Table of Contents

Table of Contents
Pivotal Cloud Foundry Documentation 11
PCF v2.5 Feature Highlights 12
Pivotal Cloud Foundry Release Notes 15
PCF v2.5 Breaking Changes 16
PCF Ops Manager v2.5 Release Notes 19
Pivotal Application Service v2.5 Release Notes 25
PAS for Windows v2.5 Release Notes 34
PCF Isolation Segment v2.5 Release Notes 36
Stemcell Release Notes 39
Stemcell (Linux) Release Notes 41
Tiles Using Xenial Stemcells in PCF 86
Architecture and Installation Overview 87
Preparing Your Firewall 90
IaaS Permissions Guidelines 92
Installing PCF in Airgapped Environments 93
Installing Pivotal Cloud Foundry on AWS 97
AWS Permissions Guidelines 101
Installing PCF on AWS Manually 102
Configuring BOSH Director on AWS 125
Installing PCF on AWS using Terraform 145
Deploying Ops Manager on AWS Using Terraform 146
Configuring BOSH Director on AWS Using Terraform 150
Deleting PCF from AWS 170
Creating a Proxy ELB for Diego SSH 174
Upgrading Ops Manager Director on AWS 177
Installing Pivotal Cloud Foundry on Azure 180
Installing PCF on Azure Manually 183
Preparing to Deploy Ops Manager on Azure Manually 184
Deploying Ops Manager on Azure Manually 189
Configuring BOSH Director on Azure Manually 198
Installing PCF on Azure Using Terraform 220
Deploying Ops Manager on Azure Using Terraform 221
Configuring BOSH Director on Azure Using Terraform 224
Troubleshooting PCF on Azure 246
Deleting a PCF on Azure Installation 249
Upgrading BOSH Director on Azure 250
Installing Pivotal Cloud Foundry on GCP 254
Recommended GCP Quotas 257
Installing PCF on GCP Manually 258
Preparing to Deploy Ops Manager on GCP Manually 259
Deploying Ops Manager on GCP Manually 280
Configuring BOSH Director on GCP Manually 283
Installing PCF on GCP Using Terraform 300
Deploying Ops Manager on GCP Using Terraform 301
Configuring BOSH Director on GCP Using Terraform 306
Configuring a Shared VPC on GCP 322

© Copyright Pivotal Software Inc, 2013-2019
Restoring PCF from Backup with BBR
Enabling External Blobstore Backups
Troubleshooting BBR
Using Ops Manager
Using the Ops Manager API
Configuring the BOSH Director with the Ops Manager API
Configuring a Product with the Ops Manager API
Using the Ops Manager UI
Using the Ops Manager Interface
Reviewing Pending Product Changes
Adding and Deleting Products
Importing and Managing Stemcells
Logging In to Apps Manager
Managing Errands in Ops Manager
Creating UAA Clients for BOSH Director
Using Your Own Load Balancer
Creating and Managing Ops Manager User and Client Accounts
Configuring Role-Based Access Control (RBAC) in Ops Manager
Creating New PAS User Accounts
Adding Existing SAML or LDAP Users to a PCF Deployment
Modifying Your Ops Manager Installation and Product Template Files
Monitoring PCF VMs from Ops Manager
Advanced Features in Ops Manager
Managing Ops Manager Verifiers
Managing Custom VM Extensions
Ops Manager Fields That Lock On Deploy
PAS Concepts
Cloud Foundry Overview
How Apps Are Staged
App Container Lifecycle
High Availability in Cloud Foundry
How Cloud Foundry Maintains High Availability
Orgs, Spaces, Roles, and Permissions
Cloud Foundry Security
Container Security
Container-to-Container Networking
Application Security Groups
GrootFS Disk Usage
Cloud Foundry Components
Cloud Controller
Messaging (NATS)
Gorouter
User Account and Authentication (UAA) Server
Garden
HTTP Routing
Diego Components and Architecture
Application SSH Components and Processes
How Diego Balances App Processes
PCF Operator Guide
Deploying Service Mesh (Beta)
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAS Network Architecture</td>
<td>790</td>
</tr>
<tr>
<td>Load Balancer</td>
<td>790</td>
</tr>
<tr>
<td>Router</td>
<td>790</td>
</tr>
<tr>
<td>Identifying the API Endpoint for Your PAS Instance</td>
<td>791</td>
</tr>
<tr>
<td>Configuring SSL/TLS Termination at HAProxy</td>
<td>792</td>
</tr>
<tr>
<td>Configuring Frontend Idle Timeout for Gorouter and HAProxy</td>
<td>794</td>
</tr>
<tr>
<td>Configuring Proxy Settings for All Applications</td>
<td>795</td>
</tr>
<tr>
<td>Restricting App Access to Internal PCF Components</td>
<td>797</td>
</tr>
<tr>
<td>Configuring Application Security Groups for Email Notifications</td>
<td>801</td>
</tr>
<tr>
<td>Configuring SSH Access for PCF</td>
<td>804</td>
</tr>
<tr>
<td>Securing Services Instance Credentials with Runtime CredHub</td>
<td>806</td>
</tr>
<tr>
<td>Identifying PAS Jobs Using vCenter</td>
<td>809</td>
</tr>
<tr>
<td>Configuring Logging in PAS</td>
<td>811</td>
</tr>
<tr>
<td>Configuring UAA Password Policy</td>
<td>817</td>
</tr>
<tr>
<td>Managing Internal MySQL for PAS</td>
<td>819</td>
</tr>
<tr>
<td>Scaling Internal MySQL for PAS</td>
<td>820</td>
</tr>
<tr>
<td>Running mysql-diag</td>
<td>823</td>
</tr>
<tr>
<td>Recovering From MySQL Cluster Downtime</td>
<td>826</td>
</tr>
<tr>
<td>Configuring Authentication and Enterprise SSO for PAS</td>
<td>833</td>
</tr>
<tr>
<td>Configuring ADFS as an Identity Provider</td>
<td>840</td>
</tr>
<tr>
<td>Configuring CA as an Identity Provider</td>
<td>842</td>
</tr>
<tr>
<td>Configuring PingFederate as an Identity Provider</td>
<td>845</td>
</tr>
<tr>
<td>Switching Application Domains</td>
<td>846</td>
</tr>
<tr>
<td>Scaling PAS</td>
<td>849</td>
</tr>
<tr>
<td>Using Docker Registries</td>
<td>853</td>
</tr>
<tr>
<td>Configuring Cell Disk Cleanup Scheduling</td>
<td>857</td>
</tr>
<tr>
<td>Custom Branding Apps Manager</td>
<td>860</td>
</tr>
<tr>
<td>Reporting App, Task, and Service Instance Usage</td>
<td>865</td>
</tr>
<tr>
<td>Reporting Instance Usage with Apps Manager</td>
<td>871</td>
</tr>
<tr>
<td>Providing a Certificate for Your TLS Termination Point</td>
<td>873</td>
</tr>
<tr>
<td>Enabling Volume Services</td>
<td>875</td>
</tr>
<tr>
<td>Rotating Runtime CredHub Encryption Keys</td>
<td>878</td>
</tr>
<tr>
<td>Administrating PAS</td>
<td>880</td>
</tr>
<tr>
<td>Managing Custom Buildpacks</td>
<td>881</td>
</tr>
<tr>
<td>Using Docker in Cloud Foundry</td>
<td>883</td>
</tr>
<tr>
<td>Creating and Managing Users with the cf CLI</td>
<td>885</td>
</tr>
<tr>
<td>Creating and Managing Users with the UAA CLI (UAAC)</td>
<td>887</td>
</tr>
<tr>
<td>Creating and Modifying Quota Plans</td>
<td>893</td>
</tr>
<tr>
<td>Using Metadata</td>
<td>897</td>
</tr>
<tr>
<td>Getting Started with the Notifications Service</td>
<td>903</td>
</tr>
<tr>
<td>Configuring Container-to-Container Networking</td>
<td>906</td>
</tr>
<tr>
<td>Administrating Dynamic Egress Policies (Beta)</td>
<td>909</td>
</tr>
<tr>
<td>Managing Isolation Segments</td>
<td>915</td>
</tr>
<tr>
<td>Routing for Isolation Segments</td>
<td>919</td>
</tr>
<tr>
<td>Using Feature Flags</td>
<td>925</td>
</tr>
<tr>
<td>Stopping and Starting Virtual Machines</td>
<td>927</td>
</tr>
<tr>
<td>Examining GrootFS Disk Usage</td>
<td>930</td>
</tr>
<tr>
<td>Enabling IPv6 for Hosted Applications</td>
<td>935</td>
</tr>
<tr>
<td>Securing Traffic into Cloud Foundry</td>
<td>936</td>
</tr>
<tr>
<td>Enabling TCP Routing</td>
<td>945</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
Troubleshooting app Deployment and Health | 1145
SSH for Apps and Services | 1150
App SSH Overview | 1151
Accessing Apps with SSH | 1153
Accessing Services with SSH | 1158
Routes and Domains | 1160
Configuring Routes and Domains | 1161
Using Weighted Routing (Beta) | 1172
Configuring CF to Route Traffic to Apps on Custom Ports | 1176
Managing Services | 1178
Managing Service Instances with the cf CLI | 1179
Sharing Service Instances | 1184
Delivering Service Credentials to an App | 1186
Managing Service Keys | 1189
Configuring Play Framework Service Connections | 1191
Using an External File System (Volume Services) | 1192
User-Provided Service Instances | 1198
Streaming App Logs | 1200
Streaming App Logs to Log Management Services | 1201
Service-Specific Instructions for Streaming App Logs | 1204
Streaming App Logs to Splunk | 1211
Streaming App Logs with Fluentd | 1213
Streaming App Logs to Azure OMS Log Analytics (Beta) | 1214
Managing Apps with the cf CLI | 1218
Running Tasks | 1219
Scaling an App Using cf scale | 1222
Using Application Health Checks | 1223
Cloud Foundry Environment Variables | 1226
Cloud Controller API Client Libraries | 1234
Considerations for Designing and Running an App in the Cloud | 1235
PCF Security and Compliance Guide | 1238
Security Processes | 1239
Pivotal Cloud Foundry Security Overview and Policy | 1240
PCF Testing, Release, and Security Lifecycle | 1242
Security Concepts | 1245
PCF Infrastructure Security | 1246
Stemcell Security | 1247
Linux Stemcell Hardening | 1248
Windows Stemcell Hardening | 1252
Certificates and TLS in PCF | 1253
TLS Connections in PCF | 1254
Custom Certificate Authorities | 1261
Managing Certificates with the Ops Manager API | 1262
Rotating Certificates | 1264
Trusted System Certificates | 1273
Disk Encryption | 1274
Security Event Logging | 1277
Network Security | 1283
BOSH DNS Network Communications | 1284
Cloud Controller Network Communications | 1287
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tips for PHP Developers</td>
<td>1421</td>
</tr>
<tr>
<td>Getting Started Deploying PHP Apps</td>
<td>1422</td>
</tr>
<tr>
<td>PHP Buildpack Configuration</td>
<td>1425</td>
</tr>
<tr>
<td>Composer</td>
<td>1429</td>
</tr>
<tr>
<td>Sessions</td>
<td>1431</td>
</tr>
<tr>
<td>New Relic</td>
<td>1432</td>
</tr>
<tr>
<td>Python Buildpack</td>
<td>1433</td>
</tr>
<tr>
<td>R Buildpack</td>
<td>1437</td>
</tr>
<tr>
<td>Ruby Buildpack</td>
<td>1439</td>
</tr>
<tr>
<td>Tips for Ruby Developers</td>
<td>1442</td>
</tr>
<tr>
<td>Getting Started Deploying Ruby Apps</td>
<td>1448</td>
</tr>
<tr>
<td>Getting Started Deploying Ruby Apps</td>
<td>1449</td>
</tr>
<tr>
<td>Getting Started Deploying Ruby on Rails Apps</td>
<td>1454</td>
</tr>
<tr>
<td>Configure Rake Tasks for Deployed Apps</td>
<td>1457</td>
</tr>
<tr>
<td>Environment Variables Defined by the Ruby Buildpack</td>
<td>1458</td>
</tr>
<tr>
<td>Configure Service Connections for Ruby</td>
<td>1459</td>
</tr>
<tr>
<td>Support for Windows Gemfiles</td>
<td>1462</td>
</tr>
<tr>
<td>Staticfile Buildpack</td>
<td>1463</td>
</tr>
<tr>
<td>Customizing and Developing Buildpacks</td>
<td>1468</td>
</tr>
<tr>
<td>Creating Custom Buildpacks</td>
<td>1469</td>
</tr>
<tr>
<td>Packaging Dependencies for Offline Buildpacks</td>
<td>1472</td>
</tr>
<tr>
<td>Merging from Upstream Buildpacks</td>
<td>1475</td>
</tr>
<tr>
<td>Upgrading Dependency Versions</td>
<td>1476</td>
</tr>
<tr>
<td>Using CI for Buildpacks</td>
<td>1481</td>
</tr>
<tr>
<td>Releasing a New Buildpack Version</td>
<td>1482</td>
</tr>
<tr>
<td>Updating Buildpack-Related Gems</td>
<td>1484</td>
</tr>
<tr>
<td>Services</td>
<td>1485</td>
</tr>
<tr>
<td>Overview</td>
<td>1486</td>
</tr>
<tr>
<td>Managing Service Brokers</td>
<td>1488</td>
</tr>
<tr>
<td>Managing Access to Service Plans</td>
<td>1492</td>
</tr>
<tr>
<td>Dashboard Single Sign-On</td>
<td>1495</td>
</tr>
<tr>
<td>Example Service Brokers</td>
<td>1499</td>
</tr>
<tr>
<td>Binding Credentials</td>
<td>1500</td>
</tr>
<tr>
<td>Enabling Service Instance Sharing</td>
<td>1502</td>
</tr>
<tr>
<td>Application Log Streaming</td>
<td>1504</td>
</tr>
<tr>
<td>Route Services</td>
<td>1505</td>
</tr>
<tr>
<td>Supporting Multiple Cloud Foundry Instances</td>
<td>1511</td>
</tr>
<tr>
<td>Logging and Metrics</td>
<td>1512</td>
</tr>
<tr>
<td>Overview of Logging and Metrics</td>
<td>1513</td>
</tr>
<tr>
<td>Loggregator Architecture</td>
<td>1515</td>
</tr>
<tr>
<td>Loggregator Guide for Cloud Foundry Operators</td>
<td>1517</td>
</tr>
<tr>
<td>Deploying a Nozzle to the Loggregator Firehose</td>
<td>1519</td>
</tr>
<tr>
<td>Installing the Loggregator Firehose Plugin for cf CLI</td>
<td>1525</td>
</tr>
<tr>
<td>Customizing Platform Log Forwarding</td>
<td>1527</td>
</tr>
<tr>
<td>Configuring the Metric Registrar</td>
<td>1529</td>
</tr>
<tr>
<td>Registering Custom App Metrics</td>
<td>1531</td>
</tr>
<tr>
<td>Identifying the Source Deployment of Metrics</td>
<td>1534</td>
</tr>
<tr>
<td>Troubleshooting and Diagnostics</td>
<td>1535</td>
</tr>
<tr>
<td>Diagnosing Problems in PCF</td>
<td>1536</td>
</tr>
<tr>
<td>Troubleshooting Problems in PCF</td>
<td>1541</td>
</tr>
</tbody>
</table>
Advanced Troubleshooting with the BOSH CLI ........................................ 1546
Troubleshooting Slow Requests in Cloud Foundry .................................. 1553
Troubleshooting TCP Routes .................................................................... 1556
Troubleshooting Router Error Responses ................................................ 1559
Troubleshooting Ops Manager for VMware vSphere ................................. 1561
Welcome to Pivotal Cloud Foundry Documentation!

Pivotal Cloud Foundry (PCF) is a cloud-native platform for deploying and operating apps. Some people install and run the platform, others develop apps on it, and others manage the development teams or make sure that the platform is secure.

The guides linked below explain how you can do what you want with Pivotal Cloud Foundry (PCF) based on your role. Or see Getting Started with PCF for a general introduction to PCF and how it works.

### PCF Guides

<table>
<thead>
<tr>
<th>I want to…</th>
<th>…which means my role is…</th>
<th>…so I should see…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read about PCF and its main features and try a simple deployment on my own local machine</td>
<td>New User</td>
<td>Getting Started with PCF</td>
</tr>
<tr>
<td>Assess PCF, plan a PCF deployment, install and configure a new PCF deployment</td>
<td>Architect</td>
<td>PCF Architecture and Installation Guide</td>
</tr>
<tr>
<td>Upgrade, monitor, back up, troubleshoot, and generally maintain an existing PCF deployment</td>
<td>Operator</td>
<td>PCF Operator Guide</td>
</tr>
<tr>
<td>Build applications that I can deploy, manage, and scale using the PAS platform.</td>
<td>Developer</td>
<td>PAS Developer Guide</td>
</tr>
<tr>
<td>Make sure that my enterprise’s PCF installation is secure and up-to-date with the latest security best practices.</td>
<td>Security Professional</td>
<td>PCF Security Guide</td>
</tr>
<tr>
<td>Manage Pivotal Application Service (PAS) users, orgs, spaces, and resources.</td>
<td>PAS Admin or Manager</td>
<td>PAS Administrator Guide</td>
</tr>
<tr>
<td>Build a “tile” that lets PCF developers use my software services in their apps.</td>
<td>PCF Tile Developer</td>
<td>PCF Tile Developer Guide</td>
</tr>
</tbody>
</table>
PCF v2.5 Feature Highlights

This topic highlights important new features included in Pivotal Cloud Foundry (PCF) v2.5.

PCF Ops Manager Highlights

Ops Manager v2.5 includes the following important major features:

Filenames for Ops Manager Release Components Have Changed on Pivotal Network

All PCF pipelines, or tools similar to them, must be updated with the new naming patterns or the tools will not detect the new versions and no upgrade will appear to be available.

Pivotal recommends updating the naming patterns in PCF pipelines or any similar pipeline tools that affect upgrades.

For more information about this change, see Ops Manager Release Artifacts Renamed on Pivotal Network.

To read the entire Breaking Changes list for PCF v2.5, see PCF v2.5 Breaking Changes.

View a Summary of Proposed Changes to the Manifest

The Pending Changes page in Ops Manager shows staged changes to the manifest compared to the state of the manifest in the last successful deployment. This allows you to easily review and understand changes to the manifest before deploying.

For more information, see About Pending Changes for a Specific Tile.

Fully Automated Use of Ops Manager

You can completely automate Ops Manager when SAML/LDAP is enabled. Configure a pre-created client account to manage automation tasks and scripts. Previously, you had to manually log in to create a client for automated use of Ops Manager.

For more information about creating Ops Manager users and clients, see Creating and Managing Ops Manager User and Client Accounts.

Support for Microsoft Azure Availability Zones

New installations of Ops Manager include support for Microsoft Azure Availability Zones (AZs). Compared to Azure Availability Sets, AZs provide higher availability, less latency, and more efficient resource management.

For more information, see Configuring BOSH Director on Azure and Configuring BOSH Director on Azure Using Terraform.

NATS Certificate Authority Rotation Through the Ops Manager API

Ops Manager allows you to regenerate and rotate the NATS certificate authority (CA) simultaneously with the Ops Manager root CA.

This feature pairs the lifecycle of the NATS CA with the Ops Manager root CA, meaning the NATS CA and Ops Manager root CA are created, activated, and deleted at the same time.

For more information about generating and rotating certificates, see Managing Certificates.

Custom VM Extensions for the BOSH Director

You can define custom VM extensions for BOSH Director VMs using the Ops Manager API. VM extensions allow you to specify IaaS-specific configurations for your VMs, such as associated security groups and load balancers.
Pivotal Application Service (PAS) Highlights

PAS v2.5 includes the following important major features:

**Service Mesh Routing Plane (Beta)**

An optional service mesh routing plane enables weighted routing, which gives you greater control over the percentage of traffic forwarded to each app mapped to a given route.

To deploy the service mesh routing plane, navigate to the Networking - Service Mesh pane of the PAS tile and select Enable under Service Mesh (Beta).

For more information, see the following topics:

- Service Mesh (Beta)
- Using Weighted Routing (Beta)

**Support for Apps That Require Multiple Ports**

PAS v2.5 supports apps that require multiple ports and apps that require custom ports. This update allows additional apps, such as Docker images and apps with TCP protocols that require multiple ports, to run on PAS.

For more information, see Configuring CF to Route Traffic to Apps on Custom Ports.

**Configure App Policies Across Spaces Using the cf CLI**

You can use the Cloud Foundry Command Line Interface (cf CLI) to configure app policies across spaces in your PCF foundation. This allows you to quickly manage and configure app policies without using the Policy Server External API.

For more information about configuring app policies across spaces, see Configuring Container-to-Container Networking.

**Disable Networking Policies**

You can disable Silk network policy enforcement for all apps in a foundation. This allows apps to send network traffic to all other apps without the need for the operator to manually configure several networking rules, which can be time-consuming.

For more information about Silk, see silk in GitHub.

**Add Metadata to Apps, Orgs, and Spaces**

You can add labels and annotations to elements throughout PAS, including apps, orgs, spaces, and more. Metadata, such as environment information, billing codes, and points of contact, make it easier to operate, monitor, and audit PAS deployments.

For more information about using metadata in PAS, see Using Metadata.

Apps Manager Highlights

Apps Manager v2.5 includes the following important major features:

**Support for Multiple Foundation Deployments (Beta)**

Apps Manager includes support for deployments with multiple PCF foundations. This allows you to search, view, and manage orgs, spaces, apps, and
service instances across multiple foundations from a single interface in Apps Manager.

**Support for Asynchronous Bindings**

Apps Manager supports asynchronous bindings. Asynchronous bindings help avoid timeouts for synchronous operations in PAS by allowing service brokers more time to complete a bind.

For more information about managing apps with Apps Manager, see [Managing Apps and Service Instances Using Apps Manager](#).
Pivotal Cloud Foundry Release Notes

Pivotal Cloud Foundry is certified by the Cloud Foundry Foundation for 2019.

Read more about the certified provider program and the requirements of providers.

This topic provides links to the release notes for Pivotal Cloud Foundry (PCF). Release notes include new features, breaking changes, bug fixes, and known issues.

To find release notes for PCF services, navigate to the documentation for the desired service from Pivotal Documentation.

PCF Release Notes

- PCF v2.5 Breaking Changes
- PCF Ops Manager v2.5 Release Notes

PCF Runtime Release Notes

- Pivotal Application Service v2.5 Release Notes
- PAS for Windows v2.5 Release Notes
- PCF Isolation Segment v2.5 Release Notes
- Pivotal Container Service (PKS) v1.2 Release Notes

Stemcell Release Notes

- Stemcell Release Notes
PCF v2.5 Breaking Changes

This topic describes the breaking changes you need to be aware of when upgrading to Pivotal Cloud Foundry (PCF) v2.5. For more information about important preparation steps you must follow before beginning an upgrade, see Upgrading Pivotal Cloud Foundry.

Critical Breaking Changes in PCF v2.5

See the following critical breaking changes before installing or upgrading to PCF v2.5:

- Deprecation of cflinuxfs2
- Filenames for Ops Manager Release Components Have Changed on Pivotal Network

cflinuxfs2 is Unsupported

Operators cannot upgrade the platform to PAS 2.5 until all app instances (AIs) are using cflinuxfs3.

During upgrade, you must check a box to verify that all AIs have been migrated. If you check the box and the AIs have not been migrated, the upgrade causes all AIs to fail.

Pivotal recommends that you communicate the change of stack to your development teams and have them migrate to cflinux3 before upgrading to v2.5.

For more information about this change, see PAS Only Supports cflinux3.

Filenames for Ops Manager Release Components Have Changed on Pivotal Network

All PCF pipelines, or tools similar to them, must be updated with the new naming patterns or the tools do not detect the new versions and no upgrade appears available.

Pivotal recommends updating the naming patterns in PCF pipelines or any similar pipeline tools that affect upgrades.

For more information about this change, see Ops Manager Release Artifacts Renamed on Pivotal Network.

Pivotal Application Service (PAS)

See the following PAS breaking changes:

Removed Logging System Property

PAS v2.5 removes the Syslog Drain Buffer Size checkbox from the Logging System pane of the PAS tile. This field corresponds to a deprecated property that does not have any affect on the logging system when configured.

It also removes the underlying property, doppler.message_drain_buffer_size. If you have any scripts that reference this property, they may fail.

Removed Log Cache Property

PAS v2.4 removed the Enable log-cache checkbox from the Advanced Features section of the PAS tile.

PAS v2.5 removes the underlying property, enable_log_cache. If you have any scripts that reference this property, they may fail.

Removed Option to Override Deployment Name to CF

PAS v2.4 introduced the use of cfGUID as the value for the deployment in metrics. For more information, see Uniquely Identify Metrics by Tile.
PAS v2.4 also included the option to override this value back to `cf`, which was used in previous versions. However, PAS v2.5 removes this override option.

To upgrade successfully, the **Use "cf" as deployment name in emitted metrics instead of unique name** option in the **Advanced Features** pane of the PAS tile must be deselected.

If you have scripts that rely on metrics with `cf` as the value for `deployment`, your scripts may break when upgrading to PAS v2.5.

Additionally, if you have PCF Healthwatch installed and you changed the value of **Use "cf" as deployment name in emitted metrics instead of unique name** in PAS, you must run the **Push Monitoring Components** errand for Healthwatch to detect the change.

### Removed Unused Apps Manager UAA Client and Property

PAS v2.5 removes an unused Apps Manager UAA client identified as `portal` and an unused refresh token property called `uaa.apps_manager_refresh_token_lifetime`.

If you have platform automation scripts, you may need to update them if they reference these properties.

### Apps Manager Verifies SSL

Starting in PAS v2.4.3, Apps Manager verifies SSL certificates for endpoints to which it proxies. For environments using self-signed certificates or certificates that are signed by a certificate authority that is not trusted by the BOSH Director, this may cause Apps Manager to show no content.

If you are upgrading to PAS v2.5 from PAS v2.4.2 or earlier, you can avoid this issue using the instructions in the following article from the Pivotal Knowledge Base: [Apps Manager shows no content due to SSL validation issue](https://kb.pivotal.io/kb acids born).

### Removed Traffic Controller Container Metrics Endpoint

PAS v2.5 removes the Traffic Controller `containermetrics/app-guid` endpoint. Any scripts that reference the `containermetrics/app-guid` endpoint may fail.

The `containermetrics/app-guid` endpoint provided access to container metrics from Traffic Controller for a specific app. In PAS v2.5, you can access container metrics through Log Cache. Log Cache includes API endpoints and a CLI plugin to query and filter logs and metrics.

To download the Log Cache Cloud Foundry Command Line Interface (cf CLI) plugin, see [cf CLI Plugins](https://github.com/pivotal-cf). For information about using the Log Cache API, see [Log Cache on GitHub](https://github.com/pivotal-cf).

### Removed dns_servers Property Breaks Platform Automation Scripts

In PAS v2.4, the **DNS Servers** field was removed from the PAS UI, but the `dns_servers` property was not removed so as not to break Platform Automation scripts. In PAS v2.5, the `dns_servers` property is removed, which breaks Platform Automation scripts if that parameter is still configured for your tiles.

### Pivotal Application Service (PAS) for Windows

#### Removed Option to Override Deployment Name

Beginning with PAS for Windows v2.4, metrics emitted by the tile are tagged with a GUID. This allows you to identify the source tile of the metrics.

For compatibility with older versions of PAS for Windows, PAS v2.4 included the option to override this value with `cf`.

PAS for Windows v2.5 removes this override option.

To upgrade from PAS for Windows v2.4 to v2.5 successfully, you must deselect the **Use "cf" as deployment name in emitted metrics instead of unique name** option in the **Advanced Features** pane of the PAS for Windows v2.4 tile.

If you have scripts that rely on metrics with `cf` as the value for `deployment`, your scripts may break when upgrading to PAS for Windows v2.5.

If you have PCF Healthwatch installed and you changed the value of **Use "cf" as deployment name in emitted metrics instead of unique name** in PAS for Windows, you must run the **Push Monitoring Components** errand to allow Healthwatch to detect this change.
Deprecation of the windows2016 Stack

The windows2016 stack is deprecated in favor of the windows stack.

PAS for Windows v2.6 will remove support entirely for the windows2016 stack.

You will need to change your apps to use the windows stack before upgrading to PAS for Windows v2.6.

PCF Ops Manager

See the following PCF Ops Manager breaking changes:

New Installations of Ops Manager v2.5 Use Azure Availability Zones by Default

New installations of Ops Manager v2.5 on Azure use Availability Zones (AZs) by default, not Availability Sets. Before deploying, you can choose either AZs or Availability Sets as your desired availability mode. If you choose AZs, you must also assign specific zones to each product. You cannot switch from AZs to Availability Sets after a successful deployment.

If you use automation tools to install Ops Manager, you may have to change the behavior of those tools to accommodate for the addition of AZs to your Azure deployment. If you are upgrading from 2.4, however, the availability mode is set to Availability Sets and cannot be changed. In that case, your automation tools may continue to work without modification.

For more information about Azure AZs, see Microsoft Azure Availability Zones Available.

Ops Manager Release Artifacts Renamed on Pivotal Network

Starting in Ops Manager v2.5, the filenames for Ops Manager release artifacts on Pivotal Network are restructured.

For example:

<table>
<thead>
<tr>
<th>Ops Manager version</th>
<th>Release artifact name structure</th>
<th>Example release artifact name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>v2.5</td>
<td>ops-manager-IAAS-VERSION-build.BUILD-NUMBER.pdf</td>
<td>ops-manager-aws-2.5.0-build.123.pdf</td>
<td>The filename lists the IaaS type after the product name and includes a patch version designator after the major and minor versions.</td>
</tr>
<tr>
<td>v2.4 and earlier</td>
<td>ops-managerVERSION-build.BUILD-NUMBER-on-IAAS.pdf</td>
<td>OpsManager2.4-build.122onAWS.pdf</td>
<td>The filename does not include a patch version designator and lists the IaaS type after the build number.</td>
</tr>
</tbody>
</table>

If you use automation scripts or other tools to access release artifacts, you may need to update them to match the new filename scheme. If your automation tools use precisely typed filenames or structures from previous versions of Ops Manager, you must update the tools to reflect the new filename structures or the tools will break.

For more information, see Ops Manager 2.5 changing the file naming scheme might break the pipeline jobs in the Pivotal Knowledge Base.

PCF Isolation Segment

See the following PCF Isolation Segment breaking changes:

Removed dns_servers Property Breaks Platform Automation Scripts

In PAS v2.4, the DNS Servers field was removed from the PAS UI, but the dns_servers property was not removed so as not to break Platform Automation scripts. In PAS v2.5, the dns_servers property is removed, which breaks Platform Automation scripts if that parameter is still configured for your tiles.
PCF Ops Manager v2.5 Release Notes

Pivotal Cloud Foundry is certified by the Cloud Foundry Foundation for 2019.

Read more about the certified provider program and the requirements of providers.

How to Upgrade

The Upgrading Pivotal Cloud Foundry topic contains instructions for upgrading to Pivotal Cloud Foundry (PCF) Ops Manager v2.5.

Releases

warning: Before installing or upgrading to Ops Manager v2.5, review the Critical Breaking Changes in PCF v2.5.

2.5.2

- This patch contains no new features or fixes. It updates Ops Manager components to address dependency requirements for other products.

Ops Manager v2.5.2 uses the following component versions:

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager</td>
<td>2.5.2-build.172*</td>
</tr>
<tr>
<td>Stemcell</td>
<td>250.17</td>
</tr>
<tr>
<td>BBR SDK</td>
<td>1.14.0</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>268.6.0</td>
</tr>
<tr>
<td>BOSH DNS</td>
<td>1.10.0</td>
</tr>
<tr>
<td>Metrics Server</td>
<td>0.0.22</td>
</tr>
<tr>
<td>CredHub</td>
<td>2.1.5*</td>
</tr>
<tr>
<td>Syslog</td>
<td>11.4.0</td>
</tr>
<tr>
<td>Windows Syslog</td>
<td>1.0.3</td>
</tr>
<tr>
<td>UAA</td>
<td>71.0</td>
</tr>
<tr>
<td>BPM</td>
<td>1.0.4</td>
</tr>
<tr>
<td>Networking</td>
<td>9</td>
</tr>
<tr>
<td>OS Conf</td>
<td>20.0.0</td>
</tr>
<tr>
<td>AWS CPI</td>
<td>73</td>
</tr>
<tr>
<td>Azure CPI</td>
<td>35.4.0</td>
</tr>
<tr>
<td>Google CPI</td>
<td>29.0.1</td>
</tr>
<tr>
<td>OpenStack CPI</td>
<td>42</td>
</tr>
<tr>
<td>vSphere CPI</td>
<td>52.1.0</td>
</tr>
<tr>
<td>BOSH CLI</td>
<td>5.4.0</td>
</tr>
<tr>
<td>Credhub CLI</td>
<td>2.4.0</td>
</tr>
<tr>
<td>BBR CLI</td>
<td>1.5.0</td>
</tr>
</tbody>
</table>

* Components marked with an asterisk have been updated.

2.5.1

- [Security Fix]: This patch addresses CVE-2019-5418, a Rails file content disclosure vulnerability.
- [Security Fix]: This patch addresses CVE-2019-5419, a Rails vulnerability that could lead to denial of service (DoS) attacks.
- **[Security Fix]**: Ops Manager operators cannot set secrets that do not match the constraints defined by the `must_match_regex` parameter.
- **[Bug Fix]**: When a redeploy is triggered by Apply Changes and that redeploy lasts multiple days, logs are generated for that entire time.
- **[UI Improvement]**: The Pending Changes page for an on-demand service broker tile now warns you explicitly that changes made applies only to the broker and not to service instances.
- **[UI Improvement]**: The notification banner that appears when a certificate in your deployment is about to expire is updated for clarity.

Ops Manager v2.5.1 uses the following component versions:

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager</td>
<td>2.5.1-build.169*</td>
</tr>
<tr>
<td>Stemcell</td>
<td>250.17</td>
</tr>
<tr>
<td>BBR SDK</td>
<td>1.14.0</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>268.6.0</td>
</tr>
<tr>
<td>BOSH DNS</td>
<td>1.10.0</td>
</tr>
<tr>
<td>Metrics Server</td>
<td>0.0.22</td>
</tr>
<tr>
<td>CredHub</td>
<td>2.1.2</td>
</tr>
<tr>
<td>Syslog</td>
<td>11.4.0</td>
</tr>
<tr>
<td>Windows Syslog</td>
<td>1.0.3</td>
</tr>
<tr>
<td>UAA</td>
<td>71.0*</td>
</tr>
<tr>
<td>BPM</td>
<td>1.0.4*</td>
</tr>
<tr>
<td>Networking</td>
<td>9</td>
</tr>
<tr>
<td>OS Conf</td>
<td>20.0.0</td>
</tr>
<tr>
<td>AWS CPI</td>
<td>73</td>
</tr>
<tr>
<td>Azure CPI</td>
<td>35.4.0</td>
</tr>
<tr>
<td>Google CPI</td>
<td>29.0.1</td>
</tr>
<tr>
<td>OpenStack CPI</td>
<td>42</td>
</tr>
<tr>
<td>vSphere CPI</td>
<td>52.1.0</td>
</tr>
<tr>
<td>BOSH CLI</td>
<td>5.4.0</td>
</tr>
<tr>
<td>Credhub CLI</td>
<td>2.4.0*</td>
</tr>
<tr>
<td>BBR CLI</td>
<td>1.5.0*</td>
</tr>
</tbody>
</table>

* Components marked with an asterisk have been updated.

---

**2.5.0**

Ops Manager v2.5.0 uses the following component versions:

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager</td>
<td>2.5.0-build.158*</td>
</tr>
<tr>
<td>Stemcell</td>
<td>250.17*</td>
</tr>
<tr>
<td>BBR SDK</td>
<td>1.14.0</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>268.6.0</td>
</tr>
<tr>
<td>BOSH DNS</td>
<td>1.10.0</td>
</tr>
<tr>
<td>Metrics Server</td>
<td>0.0.22</td>
</tr>
<tr>
<td>CredHub</td>
<td>2.1.2</td>
</tr>
<tr>
<td>Syslog</td>
<td>11.4.0</td>
</tr>
<tr>
<td>Windows Syslog</td>
<td>1.0.3</td>
</tr>
<tr>
<td>UAA</td>
<td>69.0</td>
</tr>
<tr>
<td>BPM</td>
<td>1.0.3</td>
</tr>
<tr>
<td>Networking</td>
<td>9</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
New Features in Ops Manager v2.5

Ops Manager v2.5 includes the following major features:

Pending Changes Page Displays Staged Changes

If you have changed the configuration of a tile in your deployment, use the Pending Changes page to review the changes to the existing manifest or configs line-by-line. Access a tile’s pending changes by clicking the See Changes button on the Review Pending Changes page.

Use the output on this page to verify accuracy and comprehensiveness before redeploying Ops Manager. This page is especially useful for deployments with more than one operator or administrator, or deployments that change due to both API and UI modifications.

For more information, see About Pending Changes for a Specific Tile.

Microsoft Azure Availability Zones Available

Ops Manager operators and administrators can now configure Availability Zones (AZs). AZs provide high availability, minimal latency, and efficient resource management for deployments around the world.

AZs are only available for new installations of Ops Manager. Existing installations that are being upgraded from v2.4 must use Availability Sets.

For more information, see Configuring BOSH Director on Azure and Configuring BOSH Director on Azure Using Terraform.

Host Group Configuration Available for vSphere Availability Zones

You can associate an Availability Zone (AZ) with a host group for Ops Manager deployments on vSphere. Host groups are subsets of vSphere clusters. Since you must associate a cluster with an AZ in vSphere deployments, host group configuration can help you manage deployment resources more precisely. Enable vSphere host groups from the Create Availability Zones pane in the BOSH Director tile.

For more information, see Create Availability Zones Page in Configuring BOSH Director on vSphere.

NATS Certificate Rotation Available in the Ops Manager API

Ops Manager allows you to regenerate and rotate the NATS certificate authority (CA). The public keys and expiration dates for the NATS CA are visible at /api/v0/deployed/certificates. The certificate itself is listed as an active CA on the certificates table in the Ops Manager database.

A new NATS CA is created every time a new Ops Manager root CA is created. This pairs the lifecycle of the NATS CA with the Ops Manager root CA. The two CAs are created, activated, and deleted in unison. When an operator uses the regenerate endpoint in the Ops Manager API, new leaf certificates are generated for each client using the NATS CA. The new certificates use the active NATS CA in the database.

For more information about regenerating and rotating certificates, see Rotating Certificates.
Pre-Created Clients Available in the Ops Manager API

You can now configure a pre-created client account to manage automation tasks and scripts for Ops Manager. To do this, you will need access to the UAA Command Line Client (UAAC) and Ops Manager API.

Pivotal recommends using client accounts for automated tasks because client accounts are not bound to the same authentication protocols as user accounts. A user account that controls automated components can cause those components to fail if the account experiences inconsistent availability due to permission or authentication issues.

You can configure a pre-created client for Ops Manager either before you deploy Ops Manager for the first time, or by adding it to an existing deployment.

For more information about Ops Manager users and clients, see Creating and Managing Ops Manager User and Client Accounts.

Apply Custom VM Extensions to the BOSH Director in the Ops Manager API

You can define custom VM extensions for the BOSH Director VM by creating them in the API and then applying them to an instance group using

```
PUT
/api/v0/staged/products/PRODUCT_GUID/JOBS/JOB_GUID/resource_config
```

This endpoint works for both the BOSH Director's jobs and product tiles.

Use this endpoint to configure custom behavior for the BOSH Director VM, such as allowing the Director to read or write directly to a blobstore.

For more information about custom VM extensions, see Staged VM Extensions in the Ops Manager API documentation.

Ops Manager User Interface Text is Updated

The field names, button text, error messaging, and other textual UI elements in Ops Manager are updated. This is part of an ongoing effort to improve clarity, utility, and the user experience.

This update does not impact Ops Manager product tiles or the BOSH Director. For more information about the Ops Manager UI, see Using the Ops Manager Interface.

Store BOSH Job Credentials on tmpfs

You can choose to store credentials for BOSH jobs in temporary file storage (tmpfs). This method stores credentials in volatile memory, rather than on the persistent disk. Choosing to store credentials in tmpfs may impact your BOSH deployment's availability if the tmpfs is refreshed or loses access to the deployment. You must recreate all VMs for this change to take effect.

To enable this feature, navigate to the Director Config section of the BOSH Director tile and enable the Store BOSH Job Credentials on tmpfs (beta) checkbox.

For more information on the Director Config forms, see IaaS-Specific Deployment Guidelines.

Forward Ops Manager Debug Logs to An External Store in the Syslog Form

You can choose to forward logs generated by debugging processes to an external source by selecting a checkbox in the Syslog pane of the Ops Manager Settings page. Forwarded logs are delivered to whatever external location you choose to receive other types of logs from Ops Manager.

To forward logs, select the Forward Debug Logs checkbox in the Syslog pane. This checkbox is deselected by default, because debug logs can be excessive and require a great deal of storage capacity. If you choose to use this option, ensure you have sufficient storage to contain the generated logs.

For more information about the Ops Manager Syslog pane, see Settings Page in the Using the Ops Manager Interface topic.

Custom Rsyslog Configuration Available in Ops Manager Syslog Template

You can further customize Syslog behavior by including configuration options in the Custom rsyslog Configuration field of the Ops Manager Settings page. This optional field lets you add more refined rules for syslog configuration than are permitted in the standard settings available in the Ops Manager UI.

If you configure custom rules and also choose the Forward Debug Logs checkbox, the rules apply before log forwarding takes place.
For more information about forwarding debug logs, see Forward Ops Manager Debug Logs to An External Store in the Syslog Form.

For more information about the Ops Manager Syslog pane, see Settings Page in the Using the Ops Manager Interface topic.

Known Issues

Ops Manager’s Syslog Template is Not Supported for Service Broker Tiles
Ops Manager’s Syslog template does not propagate to service broker tiles. Tile developers with service broker tiles should continue to use their own Syslog configurations. Service broker tiles with the Syslog template configured will not display the Syslog template.

For more information about the Syslog template, see Syslog Form Template Available for Tile Authors.

Monit Inaccurately Reports Health of UAA
When BOSH Director reboots, Monit may report UAA as running, even though its process state is unhealthy. The Monit start scripts for UAA use the UAA /healthz endpoint to verify UAA is running, but the /healthz endpoint does not know there is a database requirement. Monit detects UAA is healthy, even though UAA is stuck, and never restarts it.

To fix this, run monit restart uaa to restart UAA after Postgres is running.

For more information, see Monit reports UAA running on BOSH Director when it is actually unhealthy in the Pivotal Knowledge Base.

BPM Fails to Start Job Process
Some Monit processes fail to start. The bpm.log shows an exit status of 1 with no other cause for the failure.

BOSH Process Manager (BPM) calls runC to create a new container and launch a job process. However, runC cannot start the container because there is a stale /var/vcap/bpm/runc/CONTAINER-ID/state.json, where state.json is a zero-byte file. When running runC manually to start the job process, runC has an EOF error and the exit status is 1. runC fails to read the state.json and exits.

To resolve this issue, delete the zero-byte state.json file by deleting its parent directory with the following command:

```
rm -rf /var/vcap/data/bpm/runc/MJRHG--
```

Then run monit start JOB to start the job process and review the job logs to check if there are any further failures.

For more information, see BPM fails to start job process with exit status 1 in the Pivotal Knowledge Base.

API Documentation Updates
Ops Manager v2.5 includes the following updates to the API. Some of these updates are referenced in the New Features section.

Updated Sections
The following sections of the API documentation are updated in Ops Manager v2.5:

- Setting up with an internal userstore in Setup: This section includes a new parameter:
  
  setup[precreated_client_secret]

- Setting up with SAML in Setup: This section includes a new parameter:
  
  setup[precreated_client_secret]

- Setting up with LDAP in Setup: This section includes a new parameter:
  
  setup[precreated_client_secret]
- **Updating director and IaaS properties (Experimental)** in Setup: This section includes a new parameter: `job_configuration_on_tmpfs`

- Retrieving a list of install time verifiers for the director in Staged BOSH Director: This section is renamed.
- Deployed BOSH Director (Advanced): This section is designated “advanced.”
- Fetching a deployed manifest in Deployed BOSH Director (Advanced): This section is renamed.
- Fetching a deployed cloud config in Deployed BOSH Director (Advanced): This section is renamed.

### New Sections

The following sections are new in Ops Manager v2.5:

- Upgrading a stemcell
- Syslog Configuration
- Retrieving information about a deployed product
- Retrieving information about a staged product
- Setting precreated-client secret
- **Diffs**: This section also includes new subsections:
  - Fetching the BOSH Director diff
  - Fetching a product diff
Pivotal Application Service v2.5 Release Notes

Pivotal Cloud Foundry is certified by the Cloud Foundry Foundation for 2019.

Read more about the certified provider program and the requirements of providers.

Releases

⚠️ warning: Before installing or upgrading to PAS v2.5, review the Critical Breaking Changes in PCF v2.5.

2.5.2

- [Feature] Add extension buildpack support for the binary buildpack on Windows
- [Feature] push-apps-manager job can read configured UAA SAML providers
- [Feature] Task pruning age can be configured to allow PAS users with very high numbers of tasks to reduce load on the Cloud Controller database.
- [Feature] Add R offline buildpack
- [Feature] Add NGINX offline buildpack
- [Feature Improvement] Configure Diego LRP zones in Azure to point to BOSH AZs Zone
- [Feature Improvement] Link directly to cflinuxfs3 migration documentation
- [Feature Improvement] Add support for staging Docker images from repositories using schema version 2 manifests
- [Bug Fix] Fix issue where empty or malformed certificates would cause CredHub to fail to start.
- [Bug Fix] Increase TLS Certificate verification depth in Apps Manager to allow for longer certificate chains
- [Bug Fix] Improve performance when loading the home page on a foundation with many organizations
- [Bug Fix] Remove clear input icon that was overlaid on content from search bar on IE

- Bump ubuntu-xenial stemcell to version 250.29
- Bump bpm to version 1.0.4
- Bump capi to version 1.76.6
- Bump cf-autoscaling to version 218
- Bump cflinuxfs3 to version 0.80.0
- Bump credhub to version 2.1.5
- Bump diego to version 2.27.6
- Add new release nginx-offline-buildpack at version 1.0.9
- Bump push-apps-manager-release to version 668.0.9
- Add new release r-offline-buildpack at version 1.0.7
- Bump ruby-offline-buildpack to version 1.7.36

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu-xenial stemcell</td>
<td>250.29</td>
</tr>
<tr>
<td>backup-and-restore-sdk</td>
<td>1.12.0</td>
</tr>
<tr>
<td>binary-offline-buildpack</td>
<td>1.0.31</td>
</tr>
<tr>
<td>bosh-dns-aliases</td>
<td>0.0.3</td>
</tr>
<tr>
<td>bosh-system-metrics-forwarder</td>
<td>0.0.18</td>
</tr>
<tr>
<td>bpm</td>
<td>1.0.4</td>
</tr>
<tr>
<td>capi</td>
<td>1.76.6</td>
</tr>
<tr>
<td>cf-autoscaling</td>
<td>218</td>
</tr>
<tr>
<td>cf-backup-and-restore</td>
<td>0.0.11</td>
</tr>
<tr>
<td>cf-cli</td>
<td>1.13.0</td>
</tr>
<tr>
<td>cf-networking</td>
<td>2.20.0</td>
</tr>
<tr>
<td>Component</td>
<td>Version</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>cf-syslog-drain</td>
<td>8.2</td>
</tr>
<tr>
<td>cflinuxfs3</td>
<td>0.80.0</td>
</tr>
<tr>
<td>consul-drain</td>
<td>0.3</td>
</tr>
<tr>
<td>consul</td>
<td>198</td>
</tr>
<tr>
<td>credhub</td>
<td>2.1.5</td>
</tr>
<tr>
<td>diego</td>
<td>2.27.6</td>
</tr>
<tr>
<td>dotnet-core-offline-buildpack</td>
<td>2.2.7</td>
</tr>
<tr>
<td>garden-runc</td>
<td>1.19.0</td>
</tr>
<tr>
<td>go-offline-buildpack</td>
<td>1.8.35</td>
</tr>
<tr>
<td>haproxy</td>
<td>9.4.1</td>
</tr>
<tr>
<td>istio</td>
<td>1.0.1</td>
</tr>
<tr>
<td>java-offline-buildpack</td>
<td>4.18</td>
</tr>
<tr>
<td>log-cache</td>
<td>2.1.1</td>
</tr>
<tr>
<td>loggregator-agent</td>
<td>2.3</td>
</tr>
<tr>
<td>loggregator</td>
<td>103.4</td>
</tr>
<tr>
<td>mapfs</td>
<td>1.1.4</td>
</tr>
<tr>
<td>metric-registrar</td>
<td>1.0.4</td>
</tr>
<tr>
<td>mysql-monitoring</td>
<td>9.2.0</td>
</tr>
<tr>
<td>nats</td>
<td>26</td>
</tr>
<tr>
<td>nfs-volume</td>
<td>1.7.8</td>
</tr>
<tr>
<td>nginx-offline-buildpack</td>
<td>1.0.9</td>
</tr>
<tr>
<td>nodejs-offline-buildpack</td>
<td>1.6.45</td>
</tr>
<tr>
<td>notifications-ui</td>
<td>36</td>
</tr>
<tr>
<td>notifications</td>
<td>55</td>
</tr>
<tr>
<td>php-offline-buildpack</td>
<td>4.3.72</td>
</tr>
<tr>
<td>push-apps-manager-release</td>
<td>668.0.9</td>
</tr>
<tr>
<td>push-usage-service-release</td>
<td>669.0.9</td>
</tr>
<tr>
<td>pxr</td>
<td>0.14.2</td>
</tr>
<tr>
<td>python-offline-buildpack</td>
<td>1.6.29</td>
</tr>
<tr>
<td>r-offline-buildpack</td>
<td>1.0.7</td>
</tr>
<tr>
<td>routing</td>
<td>0.187.1</td>
</tr>
<tr>
<td>ruby-offline-buildpack</td>
<td>1.7.36</td>
</tr>
<tr>
<td>silk</td>
<td>2.20.0</td>
</tr>
<tr>
<td>smb-volume</td>
<td>1.0.0</td>
</tr>
<tr>
<td>staticfile-offline-buildpack</td>
<td>1.4.40</td>
</tr>
<tr>
<td>statsd-injector</td>
<td>1.6.0</td>
</tr>
<tr>
<td>syslog</td>
<td>11.4.0</td>
</tr>
<tr>
<td>uaa</td>
<td>71.0</td>
</tr>
</tbody>
</table>

2.5.1

- **[Feature Improvement]** Add support for TCP hitless reloads in haproxy to avoid connection reset errors
- **[Feature Improvement]** Auto bump the CF CLI release included with PAS for use by errands
- **[Feature Improvement]** Add ability to enable/disable gorouter hairpinning with Bypass security checks for route service lookup. This feature has potential security concerns, but may be needed for backwards compatibility. See the Route Service Internal Lookup Considerations section of [Route Services](#) for details.
- **[Security Fix]** Invitations app rejects insecure HTTP requests
- [Bug Fix] Specify buildpack for Apps Manager and related microservice applications to remove conflicts with custom buildpacks

- [Bug Fix] Fix issue in which Apps Manager shows *Invalid User* as the username for space and organization members without usernames, such as UAA clients

- [Bug Fix] When deleting an organization in Apps Manager, the user is now redirected to the home page instead of another organization page

- [Bug Fix] Fix bug where Apps Manager crash errors were not displayed

- [Bug Fix] Fix error message that incorrectly assumed that an SSL validation error occurred on any failed connection to Apps Manager’s search server

- [Bug Fix] Remove external link icon from dropdown items on Apps Manager’s Tools page

- [Bug Fix] Fix bug where app changes pushed with -no-start does not take effect when the app was started via Apps Manager

- [Bug Fix] Fix alignment of search results in Apps Manager

- [Bug Fix] Fix failed access checks on mount for NFS volume service with some Windows NFS servers

- [Bug Fix] Fix issue that can cause the Spring Boot actuator integration with Apps Manager to stop working after a rolling deployment

- [Bug Fix] Fix feature: “Operator can specify headers to be stripped from the response by the router”

- [Bug Fix] Fix diego rep to always clean up temporary download cache directory

  Bump ubuntu-xenial stemcell to version 250.25

  Bump binary-offline-buildpack to version 1.0.31

  Bump capi to version 1.76.5

  Bump cf-autoscaling to version 217

  Bump cf-cli to version 1.13.0

  Bump cflinuxfs3 to version 0.76.0

  Bump diego to version 2.27.4

  Bump dotnet-core-offline-buildpack to version 2.2.7

  Bump garden-runc to version 1.19.0

  Bump go-offline-buildpack to version 1.6.35

  Bump java-offline-buildpack to version 4.18

  Bump nfs-volume to version 1.7.8

  Bump nodejs-offline-buildpack to version 1.6.45

  Bump php-offline-buildpack to version 4.3.72

  Bump push-apps-manager-release to version 668.0.8

  Bump python-offline-buildpack to version 1.6.29

  Bump routing to version 0.187.1

  Bump ruby-offline-buildpack to version 3.7.34

  Bump staticfile-offline-buildpack to version 1.4.40

  Bump uaa to version 71.0

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu-xenial stemcell</td>
<td>250.25</td>
</tr>
<tr>
<td>backup-and-restore-sdk</td>
<td>1.12.0</td>
</tr>
<tr>
<td>binary-offline-buildpack</td>
<td>1.0.31</td>
</tr>
<tr>
<td>bosh-dns-aliases</td>
<td>0.0.3</td>
</tr>
<tr>
<td>bosh-system-metrics-forwarder</td>
<td>0.0.18</td>
</tr>
<tr>
<td>bpm</td>
<td>1.0.3</td>
</tr>
<tr>
<td>capi</td>
<td>1.76.5</td>
</tr>
<tr>
<td>cf-autoscaling</td>
<td>217</td>
</tr>
<tr>
<td>cf-backup-and-restore</td>
<td>0.0.11</td>
</tr>
<tr>
<td>cf-cli</td>
<td>1.13.0</td>
</tr>
<tr>
<td>cf-networking</td>
<td>2.20.0</td>
</tr>
<tr>
<td>cf-smoke-tests</td>
<td>40.0.40</td>
</tr>
<tr>
<td>cf-syslog-drain</td>
<td>8.2</td>
</tr>
<tr>
<td>cflinuxfs3</td>
<td>0.76.0</td>
</tr>
</tbody>
</table>
## 2.5.0

See also:

- **Breaking Changes**
- **New Features**
- **[Breaking Change]** Users must acknowledge having migrated apps off of cflinuxfs2 prior to upgrading to PAS 2.5
- **[Breaking Change]** Remove deprecated buffer size config from the Logging System pane
- **[Breaking Change]** Remove cflinuxfs2 root filesystem and buildpacks
- **[Breaking Change]** Remove unused enable_log_cache property. See Removed Log Cache Property.
- **[Breaking Change]** Remove inactive DNS server configuration that only applied to older deployments without BOSH DNS
- **[Breaking Change]** Remove unused Apps Manager UAA client and refresh token property. See Removed Unused Apps Manager UAA Client and Property.

### Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>consul</td>
<td>198</td>
</tr>
<tr>
<td>credhub</td>
<td>2.1.2</td>
</tr>
<tr>
<td>diego</td>
<td>2.27.4</td>
</tr>
<tr>
<td>dotnet-core-offline-buildpack</td>
<td>2.2.7</td>
</tr>
<tr>
<td>garden-runc</td>
<td>1.19.0</td>
</tr>
<tr>
<td>go-offline-buildpack</td>
<td>1.8.35</td>
</tr>
<tr>
<td>haproxy</td>
<td>9.4.1</td>
</tr>
<tr>
<td>istio</td>
<td>1.0.1</td>
</tr>
<tr>
<td>java-offline-buildpack</td>
<td>4.18</td>
</tr>
<tr>
<td>log-cache</td>
<td>2.1.1</td>
</tr>
<tr>
<td>loggregator-agent</td>
<td>2.3</td>
</tr>
<tr>
<td>loggregator</td>
<td>103.4</td>
</tr>
<tr>
<td>mapfs</td>
<td>1.1.4</td>
</tr>
<tr>
<td>metric-registrar</td>
<td>1.0.4</td>
</tr>
<tr>
<td>mysql-monitoring</td>
<td>9.2.0</td>
</tr>
<tr>
<td>nats</td>
<td>26</td>
</tr>
<tr>
<td>nfs-volume</td>
<td>1.7.8</td>
</tr>
<tr>
<td>nodejs-offline-buildpack</td>
<td>1.6.45</td>
</tr>
<tr>
<td>notifications-ui</td>
<td>36</td>
</tr>
<tr>
<td>notifications</td>
<td>55</td>
</tr>
<tr>
<td>php-offline-buildpack</td>
<td>4.3.72</td>
</tr>
<tr>
<td>push-apps-manager-release</td>
<td>668.0.8</td>
</tr>
<tr>
<td>push-usage-service-release</td>
<td>669.0.9</td>
</tr>
<tr>
<td>pxc</td>
<td>0.14.2</td>
</tr>
<tr>
<td>python-offline-buildpack</td>
<td>1.6.29</td>
</tr>
<tr>
<td>routing</td>
<td>0.187.1</td>
</tr>
<tr>
<td>ruby-offline-buildpack</td>
<td>1.7.34</td>
</tr>
<tr>
<td>silk</td>
<td>2.20.0</td>
</tr>
<tr>
<td>smb-volume</td>
<td>1.0.0</td>
</tr>
<tr>
<td>staticfile-offline-buildpack</td>
<td>1.4.40</td>
</tr>
<tr>
<td>statsd-injector</td>
<td>1.6.0</td>
</tr>
<tr>
<td>syslog</td>
<td>11.4.0</td>
</tr>
<tr>
<td>uaa</td>
<td>71.0</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
- **[Breaking Change]** Remove option to override deployment name to CF. See [Removed Option to Override Deployment Name to CF](#).
- **[Breaking Change]** Remove unused jobs `mysql-feature` and `mysql-backup`.
- **[Breaking change]** Increase default and minimum CPU core count from 1 to 2 for internal blobstore VMs to improve reliability.
- **[Breaking Change]** Bump log-cache API to version 2.x which moves `/v1` endpoints to `/api/v1`
- **[Breaking Change]** Internal MySQL always uses Percona XtraDB Cluster.
- **[Feature]** Add optional Istio based routing tier (BETA). See [Optional Istio-Based Routing Tier (BETA)](#).
- **[Feature]** Add ability to disable network policy enforcement between applications. See [Disable Network Policy Enforcement Between Apps](#).
- **[Feature]** Add metric-registrar to enable App Developers to output custom application metrics that can be monitored by platform-provided tooling.
- **[Feature]** Users in `usage_service.audit` group can access the Usage Service API using their CF OAuth token.
- **[Feature Improvement]** Cloud Controller fetches application metrics from log-cache.
- **[Feature Improvement]** BPM is enabled for Credhub.
- **[Feature Improvement]** nfs-broker connects to its database using TLS.

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu-xenial stemcell</td>
<td>250.21</td>
</tr>
<tr>
<td>backup-and-restore-sdk</td>
<td>1.12.0</td>
</tr>
<tr>
<td>binary-offline-buildpack</td>
<td>1.0.30</td>
</tr>
<tr>
<td>bosh-dns-aliases</td>
<td>0.0.3</td>
</tr>
<tr>
<td>bosh-system-metrics-forwarder</td>
<td>0.0.18</td>
</tr>
<tr>
<td>bpm</td>
<td>1.0.3</td>
</tr>
<tr>
<td>capi</td>
<td>1.76.3</td>
</tr>
<tr>
<td>cf-autoscaling</td>
<td>216</td>
</tr>
<tr>
<td>cf-backup-and-restore</td>
<td>0.0.11</td>
</tr>
<tr>
<td>cf-cli</td>
<td>1.12.0</td>
</tr>
<tr>
<td>cf-networking</td>
<td>2.20.0</td>
</tr>
<tr>
<td>cf-smoke-tests</td>
<td>40.0.40</td>
</tr>
<tr>
<td>cf-syslog-drain</td>
<td>8.2</td>
</tr>
<tr>
<td>cflinuxfs3</td>
<td>0.72.0</td>
</tr>
<tr>
<td>consul-drain</td>
<td>0.0.3</td>
</tr>
<tr>
<td>consul</td>
<td>198</td>
</tr>
<tr>
<td>credhub</td>
<td>2.1.2</td>
</tr>
<tr>
<td>diego</td>
<td>2.27.0</td>
</tr>
<tr>
<td>dotnet-core-offline-buildpack</td>
<td>2.2.5</td>
</tr>
<tr>
<td>garden-runc</td>
<td>1.18.0</td>
</tr>
<tr>
<td>go-offline-buildpack</td>
<td>1.8.33</td>
</tr>
<tr>
<td>haproxy</td>
<td>9.4.1</td>
</tr>
<tr>
<td>istio</td>
<td>1.0.1</td>
</tr>
<tr>
<td>java-offline-buildpack</td>
<td>4.16.1</td>
</tr>
<tr>
<td>log-cache</td>
<td>2.1.1</td>
</tr>
<tr>
<td>loggregator-agent</td>
<td>2.3</td>
</tr>
<tr>
<td>loggregator</td>
<td>103.4</td>
</tr>
<tr>
<td>mapfs</td>
<td>1.1.4</td>
</tr>
<tr>
<td>metric-registrar</td>
<td>1.0.4</td>
</tr>
<tr>
<td>mysql-monitoring</td>
<td>9.2.0</td>
</tr>
<tr>
<td>nats</td>
<td>26</td>
</tr>
<tr>
<td>nfs-volume</td>
<td>1.7.7</td>
</tr>
<tr>
<td>nodejs-offline-buildpack</td>
<td>1.6.43</td>
</tr>
</tbody>
</table>
How to Upgrade

The procedure for upgrading to Pivotal Application Service (PAS) v2.5 is documented in the Upgrading Pivotal Cloud Foundry topic.

When upgrading to PAS v2.5, be aware of the following upgrade considerations:

- If you previously used an earlier version of PAS, you must first upgrade to PAS v2.4 to successfully upgrade to PAS v2.5.
- Some partner service tiles may be incompatible with PCF v2.5. Pivotal is working with partners to ensure their tiles are updated to work with the latest versions of PCF.

For information about which partner service releases are currently compatible with PCF v2.5, review the appropriate partners services release documentation at https://docs.pivotal.io, or contact the partner organization that produces the tile.

New Features in PAS v2.5

PAS Only Supports cflinuxfs3

cflinuxfs3 is the only stack supported in PAS v2.5. In PAS v2.4, both the cflinuxfs3 and cflinuxfs2 stacks were supported.

All apps using cflinuxfs2 must be restaged with cflinuxfs3 before upgrading to PAS v2.5. For more information, see the Migrate Apps to cflinuxfs3 of the Upgrade Preparation Checklist for PCF v2.5 topic.

The cflinuxfs3 stack is derived from Ubuntu Bionic 18.04. The cflinuxfs2 stack is derived from Ubuntu Trusty 14.04, for which support ends in April 2019. For more information, see cflinuxfs3 Release in the BOSH documentation.

Disable Network Policy Enforcement Between Apps

You can disable Silk network policy enforcement between apps in the Networking pane of the PAS tile. Disabling network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

Silk is a network fabric for containers designed for Cloud Foundry. For more information about Silk, see silk in GitHub.

Service Mesh Routing Plane (Beta)

Operators have the option to deploy a service mesh routing plane in addition to the existing HTTP and TCP routers. The first new feature provided by
service mesh is weighted routing.

For more information, see Service Mesh (Beta).

**Weighted Routing (Beta)**

In deployments with service mesh configured, developers can configure routing weights for apps. Weighted routing allows you to map multiple apps to the same route and control the amount of traffic sent to each of them. Some use cases include include A/B testing, canary releases, or rolling out features over time to incremental user-bases.

For more information, see Using Weighted Routing (Beta).

**Diego BBS Increases Routing Stability**

Diego Bulletin Board System (BBS) improves routability for application instances running on cells with unstable presences.

Previously, when the cell missed a heartbeat, application instances on unstable cells were immediately replaced and routability was terminated. Now, when the cell misses a heartbeat, the routing to the application instance is maintained in case the cell is able to regain its presence. A replacement app is scheduled in case the cell does not regain its presence.

**Support For Apps That Require Multiple Ports**

PAS v2.5 supports apps that require multiple ports and apps that require custom ports. This update allows additional apps, such as Docker images and apps with TCP protocols that require multiple ports, to run on PAS.

For more information, see Configuring CF to Route Traffic to Apps on Custom Ports.

**Configure Networking Policies Across Spaces Using the cf CLI**

You can use the Cloud Foundry Command Line Interface (cf CLI) to configure networking policies across spaces in your PCF foundation. This allows you to quickly manage and configure networking policies without using the API.

**Apps Manager Supports Asynchronous Service Bindings**

Apps Manager supports services that take advantage of the flexibility of asynchronous bindings. Asynchronous bindings provide service brokers more time to perform the operations required to complete a bind. This can help service brokers avoid timing out for synchronous operations in PAS.

When you bind an app to a service that uses asynchronous bindings, Apps Manager displays the status of the service while the bind is still pending.

For more information about asynchronous operations in services, see the Open Service Broker API documentation.

**Specify Metadata for Apps, Orgs, and Spaces**

You can specify metadata to provide additional information about apps, orgs, and spaces. Metadata can help with operating, monitoring, and auditing your PAS deployment. Example uses of metadata include environment information, billing codes, points of contact, and information about security or risk.

You can specify two types of metadata: labels and annotations. You can query objects based on labels but not annotations.

For more information about metadata and how to add, update, and view metadata, see Using Metadata.

**Generate Garden Component Logs with Binary File**

Garden, the component that PCF uses to create and manage isolated environments called containers, includes a binary file called `dontpanic`. Operators can run `>/var/vcap/packages/garden/bin/dontpanic` to generate a tar file of Garden component logs for use in troubleshooting or reporting an issue.
mysql-restore and mysql-backup Jobs Are Removed

PAS v2.5 does not use the `mysql-restore` and `mysql-backup` jobs to back up MySQL internally. Instead, each BOSH job is backed up separately with the BBR. `mysql-restore` and `mysql-backup` have therefore been removed.

For more information, see Backing Up and Restoring Pivotal Cloud Foundry.

Cloud Controller Retrieves Container Metrics from Log Cache

In PAS v2.5, Cloud Controller supports retrieving container metrics both from Traffic Controller and directly from Log Cache. When Cloud Controller retrieves container metrics directly from Log Cache, it can use the new Diego metric tags with these metric envelopes.

As for Traffic Controller, the `containermetrics/app-guid` endpoint no longer exists.

For more information about Log Cache and Traffic Controller, see Configuring Logging in PAS, Loggregator Architecture, and Overview of Logging and Metrics.

Known Issues

Extended cf logs Downtime

When upgrading from PAS v2.4, there is extended `cf logs` downtime. This causes the smoke test errand to fail during upgrade.

Scheduler Fails to Deploy If Connecting to MySQL Over TLS

Scheduler for PCF has a dependency on MySQL for PCF v2.x and currently does not support connecting to a MySQL database over TLS. If MySQL for PCF v2.x is configured to require TLS connection to the database, Scheduler fails to install.

Pivotal recommends configuring Scheduler to allow Optional TLS. This approach can be applied to other use cases, such as Spring Cloud Services, or for developer applications.

For more information, see PAS Scheduler fails to deploy in 2.5 in the Pivotal Knowledge Base.

Cloud Controller Error Causes PCF Upgrade to Fail

With buildpacks now having stack associations, additional validation must be added while upgrading to PAS v2.2 and later. This can generate a new `StacklessAndStackfulMatchingBuildpacksError` error in the post-start scripts.

For more information and instructions for resolving this issue, see Pivotal Cloud Foundry upgrade fails with a StacklessAndStackfulMatchingBuildpacksExistError Cloud Controller Error in the Pivotal Knowledge Base.

Apps Usage Service Errand Fails on Foundations With Self-Signed Certificates

The Apps Usage Service errand fails to deploy in PAS on foundations that use a self-signed or private certificate for Transport Layer Security (TLS). The Apps Usage Service errand fails with the following `curl` SSL error:

```
curl: (60) server certificate verification failed.
CAfile: /etc/ssl/certs/ca-certificates.crt
CRLfile: none
More details here: http://curl.haxx.se/docs/sslcerts.html
curl performs SSL certificate verification by default, using a "bundle" of Certificate Authority (CA) public keys (CA certs).
If the default bundle file isn't adequate, you can specify an alternate file using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in the bundle, the certificate verification probably failed due to a problem with the certificate (it might be expired, or the name might not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use the -k (or --insecure) option.
Error: failed to run job-process: exit status 1 (exit status 1)
```
This error occurs because the Apps Usage Service errand uses native `curl` instead of `cf curl` to query an API endpoint. The native `curl` command fails SSL verification in the PAS deployment.

For more information and instructions for resolving this issue, see Apps Usage errand fails in PAS 2.5 with curl SSL error.

HTTP Response Headers Field Not Functional

The Remove Specified HTTP Response Headers field is not functional. If you modify this field, the router does not remove the headers as expected. This is due to a bug that prevents the configuration from being applied to the underlying job.

For more information, see Remove Specified HTTP Response Headers in PCF does not work in the Pivotal Knowledge Base.

Azure Availability Zones Not Compatible with Diego

While PAS v2.5 supports Azure availability zones (AZs), enabling this feature is not recommended for production workloads. With Azure AZs, Diego cannot detect which zone the cells are in. This means PAS cannot distribute app instances across different AZs, causing PCF not to run in high availability mode.

For more information, see Azure Availability Zones are not compatible with Diego in PAS 2.5 in the Pivotal Knowledge Base.

Apps Manager SSL Validation Cannot Be Disabled in v2.5.0–2.5.2

In v2.5.0–2.5.2, Apps Manager ignores the Disable SSL certificate verification for this environment PAS tile setting. For environments using SSL certificates signed by an untrusted certificate authority (CA), this may cause Apps Manager to show no content.

To resolve this issue, see Apps Manager shows no content due to SSL validation issue.

Apps Manager Only Allows One Intermediate Certificate Authority in v2.5.0 and v2.5.1

In v2.5.0 and v2.5.1, Apps Manager does not accept SSL certificates that have a signing chain with more than one intermediate certificate authority between the SSL certificate and the root certificate authority. This includes certificates from backend services such as the Cloud Controller API.

If there is more than one certificate authority, Apps Manager does not show content, and the logs for Apps Manager include the text `certificate chain too long`.

If you must use an SSL certificate chain with more than one intermediate certificate authority in your environment, contact Pivotal Support to discuss options for working around this issue.

Metric Registrar App Log Error Message

An `invalid log drain` error message may appear in app logs on foundations where the Metric Registrar is enabled. This error message indicates that the current version of Loggregator cannot parse the scheme used by the Metric Registrar.

This error message does not indicate that the app logs are incorrect or incomplete. There is no action required.
PAS for Windows v2.5 Release Notes

This topic contains release notes for Pivotal Application Service (PAS) for Windows.

How to Upgrade

The PAS for Windows v2.5 tile is available with the release of Pivotal Cloud Foundry (PCF) version v2.5. To use the PAS for Windows v2.5 tile, you must install Ops Manager v2.5 or later and PAS v2.5 or later.

Releases

2.5.0

- [Feature Improvement] Enable SMB \net use -style mounting with FQDN paths on Windows 2016 cells
- [Security Fix] Rotate Diego intermediate CA before current certificate expires

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>windows2019 stemcell</td>
<td>2019.3</td>
</tr>
<tr>
<td>cf-windows-smoke-tests</td>
<td>40.0.40</td>
</tr>
<tr>
<td>diego</td>
<td>2.27.0</td>
</tr>
<tr>
<td>event-log</td>
<td>0.7.0</td>
</tr>
<tr>
<td>garden-runc</td>
<td>1.18.0</td>
</tr>
<tr>
<td>hwc-offline-buildpack</td>
<td>3.1.6</td>
</tr>
<tr>
<td>loggregator-agent</td>
<td>2.3</td>
</tr>
<tr>
<td>loggregator</td>
<td>103.4</td>
</tr>
<tr>
<td>winc</td>
<td>1.9.0</td>
</tr>
<tr>
<td>windows-utilities</td>
<td>0.11.0</td>
</tr>
<tr>
<td>windowsfs-release</td>
<td>1.0.0</td>
</tr>
</tbody>
</table>

New Features in PAS for Windows v2.5

PAS for Windows v2.5 includes the following major features:

Support for Windows Server 2019

PAS for Windows v2.5 requires Windows Server 2019. This means developers can deploy .NET and Windows applications on Windows Server 2019 in PAS. Moving to Windows Server 2019 adds stability and enhances a number of container APIs used in the Garden Windows implementation (\winc) of PAS for Windows.

Windows Server 2019 stemcells for supported public cloud providers are available for download on Pivotal Network. For vSphere stemcells, see the documentation on producing them from a Windows image provided by your Microsoft VLSC account. For more information, see Creating a vSphere Windows Stemcell.

Note: PAS for Windows v2.5 supports Windows Server 2019-based stemcells only.

.NET Apps Trust CA Certificates Configured in BOSH Director

Applications deployed on PAS for Windows now automatically trust your certificate authority (CA) certificates configured in the Security pane of the BOSH Director tile. To enable .NET apps to trust these CA certificates, PAS for Windows includes them in the system trust store of Windows containers.
For instructions on configuring trusted certificates in the BOSH Director tile, see Setting Trusted Certificates.

Deprecation of the windows2016 Stack

The windows2016 stack is deprecated in favor of the windows stack. PAS for Windows v2.6 removes support for windows2016.

You can migrate your apps from windows2016 to windows using Stack Auditor, a Cloud Foundry CLI plugin. For more information, see Using the Stack Auditor Plugin.

Known Issues

This section includes known issues with PAS for Windows v2.5.

Removed Option to Override Deployment Name to CF

PAS v2.4 introduced the use of cf-GUID as the value for the deployment in metrics. For more information, see Uniquely Identify Metrics by Tile.

PAS v2.4 also included the option to override this value back to cf, which was used in previous versions. However, PAS v2.5 removes this override option.

To upgrade successfully, the Use "cf" as deployment name in emitted metrics instead of unique name option in the Advanced Features pane of the PAS tile must be deselected.

If you have scripts that rely on metrics with cf as the value for deployment, your scripts may break when upgrading to PAS v2.5.

Additionally, if you have PCF Healthwatch installed and you changed the value of Use "cf" as deployment name in emitted metrics instead of unique name in PAS, you must run the Push Monitoring Components errand for Healthwatch to detect the change.

AWS Deployments Using CloudFront-Enabled Blobstores May Result in cf push Failing Due to Untrusted Root Certificates

Symptom

```
cf push
```
to Windows cells on Amazon Web Services (AWS) fails with the following error:

```
ERR: Downloading failed
```

Explanation

Windows Server 2019 stemcells (version v2019.2) on AWS are missing the root CA certificate that is used to sign Amazon’s CloudFront CDN certificate. The missing root CA certificate is the VeriSign Class 3 Public Primary Certification Authority.

If you deploy PAS with a blobstore that is fronted by CloudFront, there is an x509 certificate error when downloading the app droplet from the blobstore. As a result, `cf push` fails.

Workaround

Add the missing VeriSign root CA certificate to Trusted Certificates in Ops Manager to ensure that the certificate presented by CloudFront is trusted on Windows cells, allowing the droplet to be successfully downloaded from the blobstore.
PCF Isolation Segment v2.5 Release Notes

Releases

2.5.2

- [Feature Improvement] Configure Diego LRP zones in Azure to point to BOSH AZs Zone
- [Feature Improvement] Add support for staging Docker images from repositories using schema version 2 manifests
- Bump ubuntu-xenial stemcell to version 250.29
- Bump bpm to version 1.0.4
- Bump cflinuxfs3 to version 0.80.0
- Bump diego to version 2.27.6

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu-xenial stemcell</td>
<td>250.29</td>
</tr>
<tr>
<td>bpm</td>
<td>1.0.4</td>
</tr>
<tr>
<td>cf-networking</td>
<td>2.20.0</td>
</tr>
<tr>
<td>cflinuxfs3</td>
<td>0.80.0</td>
</tr>
<tr>
<td>diego</td>
<td>2.27.6</td>
</tr>
<tr>
<td>garden-runc</td>
<td>1.19.0</td>
</tr>
<tr>
<td>haproxy</td>
<td>9.4.1</td>
</tr>
<tr>
<td>loggregator-agent</td>
<td>2.3</td>
</tr>
<tr>
<td>loggregator</td>
<td>103.4</td>
</tr>
<tr>
<td>mapfs</td>
<td>1.1.4</td>
</tr>
<tr>
<td>nfs-volume</td>
<td>1.7.8</td>
</tr>
<tr>
<td>routing</td>
<td>0.187.1</td>
</tr>
<tr>
<td>silk</td>
<td>2.20.0</td>
</tr>
<tr>
<td>smb-volume</td>
<td>1.0.0</td>
</tr>
<tr>
<td>syslog</td>
<td>11.4.0</td>
</tr>
</tbody>
</table>

2.5.1

- [Feature Improvement] Add support for TCP hitless reloads in haproxy to avoid connection reset errors
- [Feature Improvement] Add ability to enable/disable gorouter hairpinning
- [Bug Fix] Fix failed access checks on mount for NFS volume service with some Windows NFS servers
- [Bug Fix] Fix feature: “Operator can specify headers to be stripped from the response by the router”
- [Bug Fix] Fix diego rep to always clean up temporary download cache directory
- Bump ubuntu-xenial stemcell to version 250.25
- Bump cflinuxfs3 to version 0.76.0
- Bump diego to version 2.27.4
- Bump garden-runc to version 1.19.0
- Bump nfs-volume to version 1.7.8
- Bump routing to version 0.187.1

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu-xenial stemcell</td>
<td>250.25</td>
</tr>
<tr>
<td>bpm</td>
<td>1.0.3</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
About PCF Isolation Segment

The PCF Isolation Segment v2.5 tile is available for installation with PCF v2.5.

Isolation segments provide dedicated pools of resources where you can deploy apps and isolate workloads. Using isolation segments separates app resources as completely as if they were in different CF deployments but avoids redundant management and network complexity.
For more information about using isolation segments in your deployment, see the Managing Isolation Segments topic.

How to Install

The procedure for installing PCF Isolation Segment v2.5 is documented in the Installing PCF Isolation Segment topic.

To install a PCF Isolation Segment, you must first install PCF v2.5.

New Features in PCF Isolation Segment v2.5

Disable Network Policy Enforcement Between Apps

You can now disable Silk network policy enforcement between apps in the Networking pane of the Isolation Segment tile. Disabling network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

Silk is a network fabric for containers designed for Cloud Foundry. For more information about Silk, see silk in GitHub.

mysql-restore and mysql-backup Jobs Are Removed

PAS v2.5 does not use the mysql-restore and mysql-backup jobs to back up MySQL internally. Instead, each BOSH job is backed up separately with the BBR. mysql-restore and mysql-backup have therefore been removed.

For more information, see Backing Up and Restoring Pivotal Cloud Foundry.

About Advanced Features

The Advanced Features section of the PCF Isolation Segment v2.5 tile includes new functionality that may have certain constraints.

Although these features are fully supported, Pivotal recommends caution when using them in production.

Known Issues

NSX-T v2.3.1 and Earlier Not Compatible with PCF Isolation Segment

NSX-T tiles v2.3.1 and earlier are not compatible with PCF Isolation Segment. The Gorouters in an Isolation Segment are not given access in the firewall rules for NSX-T v2.3.1 and earlier, which prevents them from communicating with apps.

NSX-T tiles v2.3.2 and later give access to the Gorouters in an Isolation Segment and are compatible with PCF Isolation Segment.
Stemcell Release Notes

This topic provides an overview of the Stemcell versions used by Pivotal Cloud Foundry (PCF).

Stemcells are available for download on Pivotal Network. Once downloaded, you can upload them to your deployment using the Stemcell configuration pane of a given tile or by navigating to the Stemcell Library page in Ops Manager v2.1 or later.

Before you upgrade PCF, always verify that any tiles that you have installed, either from Pivotal or a partner, are compatible with the version of PCF you are deploying and its supported stemcells. For more information on checking PCF and stemcell compatibility, see Tile Compatibility in the Upgrade Checklist.

To find out which stemcell version is used by which tile version, check the release notes or the Pivotal Network download page for the tile.

Note: A stemcell is a versioned OS image that BOSH uses to create VMs for the BOSH Director, Pivotal Application Service (PAS), and other PCF Service tiles, such as MySQL for PCF. For more information, see What is a Stemcell? in the BOSH documentation.

Ubuntu Linux Stemcell Lines

This section lists Linux-based stemcells used in Pivotal Cloud Foundry.

Trusty Stemcells

Each stemcell line has a corresponding release notes section. See the links below:

- Stemcell 3586.x Release Notes
- Stemcell 3541.x Release Notes
- Stemcell 3468.x Release Notes
- Stemcell 3445.x Release Notes
- Stemcell 3421.x Release Notes
- Stemcell 3363.x Release Notes

Trusty End-of-Support

Ubuntu 14.04 LTS (Trusty) stemcells are reaching end-of-support in April 2019. Starting in September 2018, Pivotal and its partners will begin releasing product tiles for PCF that support Ubuntu 16.04 LTS (Xenial) stemcells instead. Using supported stemcells is necessary to avoid exposure to security vulnerabilities.

Xenial Stemcells

Each stemcell line has a corresponding release notes section. See the links below:

- Stemcell 97.x Release Notes
- Stemcell 87.x Release Notes
- Stemcell 81.x Release Notes
- Stemcell 60.x Release Notes
- Stemcell 50.x Release Notes
- Stemcell 40.x Release Notes

Add-on Support for Xenial Stemcells

If you are using any of the following PCF add-ons, you must update the add-on and its configuration to be compatible with the Ubuntu Xenial 16.04 stemcell before deploying any tiles that use Xenial. The following add-on versions support the Xenial stemcell:
ClamAV Add-on for PCF v1.2.22 or later. For information on how to update this add-on, see Updating ClamAV Add-on for PCF to Run with Xenial Stemcells.

File Integrity Monitoring (FIM) Add-on for PCF v1.4.28 or later. For information on how to update this add-on, see Updating FIM Add-on for PCF to Run with Xenial Stemcells.

IPSec Add-on for PCF v1.9.9 or later. For information on how to update this add-on, see Updating IPSec Add-on for PCF to Run with Xenial Stemcells.

For information on which PCF tile releases now use Xenial, see Tiles Using Xenial Stemcells in PCF.

Windows Stemcell Lines

This section lists Windows-based stemcells used in Pivotal Cloud Foundry. Each stemcell line has a corresponding release notes section. See the links below:

- Stemcell 1709.x (Windows Server version 1709) Release Notes
- Stemcell 1200.x (Windows2012R2) Release Notes
- Stemcell 1803.x (Windows Server version 1803) Release Notes
Stemcell (Linux) Release Notes

This topic includes release notes for Linux stemcells used with Pivotal Cloud Foundry (PCF).

Xenial Stemcells

The following sections describe each Xenial stemcell release.

250.x

This section includes release notes for the 250 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

250.29

Available in Pivotal Network

Release Date: April 08, 2019

Periodic stemcell bump (Apr 9, 2019)

250.25

Available in Pivotal Network

Release Date: March 25, 2019

Periodic stemcell bump (Mar 26, 2019)

250.23

Available in Pivotal Network

Release Date: March 21, 2019

Periodic stemcell bump (Mar 22, 2019)

250.21

Available in Pivotal Network

Release Date: March 12, 2019

Periodic stemcell bump (Mar 15, 2019)

250.17

Available in Pivotal Network

Release Date: February 25, 2019

Periodic stemcell bump (Mar 06, 2019)

250.9
Release Date: February 12, 2019

Periodic Ubuntu Xenial stemcell bump (Feb 13, 2019)

250.4

Release Date: January 29, 2019

First published xenial 250. stemcell.

170.x

This section includes release notes for the 170 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

170.48

Available in Pivotal Network

Release Date: April 08, 2019

Periodic stemcell bump (Apr 8, 2019)

170.45

Available in Pivotal Network

Release Date: March 25, 2019

Periodic stemcell bump (Mar 26, 2019)

170.43

Release Date: March 21, 2019

Periodic stemcell bump (Mar 22, 2019)

170.39

Available in Pivotal Network

Release Date: March 11, 2019

Periodic stemcell bump (Mar 15, 2019)

170.38

Available in Pivotal Network

Release Date: February 25, 2019

Periodic stemcell bump (Mar 06, 2019)

170.30

Available in Pivotal Network
Release Date: February 12, 2019

Periodic Ubuntu Xenial stemcell bump (Feb 12, 2019)

170.25

Available in Pivotal Network

Release Date: January 28, 2019

Periodic Ubuntu Xenial stemcell bump (Jan 28, 2019)

170.24

Available in Pivotal Network

Release Date: January 23, 2019

Addresses "USN-3866-1: Ghostscript vulnerability"

170.23

Available in Pivotal Network

Release Date: January 22, 2019

Addresses "USN-3863-1: APT vulnerability"

170.21

Release Date: January 15, 2019

Periodic Ubuntu Xenial stemcell bump (Jan 15, 2019)

170.19

Available in Pivotal Network

Release Date: January 11, 2019

Addresses "USN-3855-1: systemd vulnerabilities"

170.15

Available in Pivotal Network

Release Date: December 20, 2018

Periodic Ubuntu Xenial stemcell bump (Dec 26, 2018)

170.14

Available in Pivotal Network

Release Date: December 17, 2018

Periodic Ubuntu Xenial stemcell bump (Dec 17, 2018)
170.13
Release Date: December 11, 2018

Fixes
- Google: hostname should always be BOSH Agent ID ([#57](#), [#16225262](#))

170.12
Release Date: December 04, 2018
Periodic Ubuntu Xenial stemcell bump (Dec 05, 2018)

170.9
Release Date: November 19, 2018
Periodic Ubuntu Xenial stemcell bump (Nov 19, 2018)

170.6
Release Date: November 15, 2018
Includes updates to address:
- [USN-3820-2](#): Linux kernel (HWE) vulnerabilities

170.5
Release Date: November 05, 2018
Periodic Ubuntu Xenial stemcell bump (Nov 05, 2018)

97.x
This section includes release notes for the 97 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

97.74
Available in Pivotal Network
Release Date: April 08, 2019
Periodic stemcell bump (Apr 8, 2019)

97.71
Available in Pivotal Network
Release Date: March 25, 2019
Periodic stemcell bump (Mar 26, 2019)
97.67
Available in Pivotal Network
Release Date: March 15, 2019
Periodic stemcell bump (Mar 22, 2019)

97.66
Available in Pivotal Network
Release Date: March 11, 2019
Periodic stemcell bump (Mar 15, 2019)

97.65
Available in Pivotal Network
Release Date: February 25, 2019
Periodic stemcell bump (Mar 06, 2019)

97.57
Available in Pivotal Network
Release Date: February 12, 2019
Periodic Ubuntu Xenial stemcell bump (Feb 13, 2019)

97.53
Available in Pivotal Network
Release Date: January 28, 2019
Periodic Ubuntu Xenial stemcell bump (Jan 28, 2019)

97.52
Available in Pivotal Network
Release Date: January 23, 2019
Addresses "USN-3866-1: Ghostscript vulnerability"

97.51
Available in Pivotal Network
Release Date: January 22, 2019
Addresses "USN-3863-1: APT vulnerability"

97.49
Release Date: January 15, 2019
Periodic Ubuntu Xenial stemcell bump (Jan 15, 2019)
97.47
Available in Pivotal Network

Release Date: January 11, 2019
Addresses “USN-3855-1: systemd vulnerabilities”
97.43
Available in Pivotal Network

Release Date: December 20, 2018
Periodic Ubuntu Xenial stemcell bump (Dec 26, 2018)
97.42
Available in Pivotal Network

Release Date: December 17, 2018
Periodic Ubuntu Xenial stemcell bump (Dec 17, 2018)
97.41
Available in Pivotal Network

- Google: hostname should always be BOSH Agent ID (
  #57 [1], #162225262 [2])
- Unpin rsyslog (was v8.22; backported from 170.x; 
  #162514665 [3])
- Periodic Ubuntu Xenial updates

Release Date: December 04, 2018
Periodic Ubuntu Xenial stemcell bump (Dec 05, 2018)
97.39
Available in Pivotal Network

Release Date: November 19, 2018
Periodic Ubuntu Xenial stemcell bump (Nov 19, 2018)
97.34
Available in Pivotal Network

© Copyright Pivotal Software Inc, 2013-2019
Available in Pivotal Network

97.32
Release Date: November 15, 2018
Includes updates to address:

- USN-3820-2: Linux kernel (HWE) vulnerabilities

97.31
Release Date: November 05, 2018
Periodic Ubuntu Xenial stemcell bump (Nov 08, 2018)
This stemcell addresses a bug introduced in the 97.31 release. The bug impacts AWS light stemcell users only.

97.22
Release Date: October 04, 2018
Addresses “USN-3777-2: Linux kernel (HWE) vulnerabilities” (Oct 04, 2018)

97.19
Release Date: October 02, 2018
(Oct 02, 2018)

97.18
Release Date: September 24, 2018
Periodic Ubuntu Xenial stemcell bump (Sep 25, 2018)

97.17
Release Date: September 18, 2018
Fixes mounting persistent disk issue with bosh-agent. (Sep 19, 2018)

97.16

© Copyright Pivotal Software Inc, 2013-2019
Available in Pivotal Network

Release Date: September 10, 2018

Periodic Ubuntu Xenial stemcell bump (Sep 11, 2018)

97.15

Release Date: August 27, 2018

Bump Ubuntu Xenial stemcells for “USN-3756-1: Intel Microcode vulnerabilities”

97.12

Release Date: August 14, 2018

Bump Ubuntu Xenial stemcells for “USN-3740-2: Linux kernel (HWE) vulnerabilities”

97.10

Release Date: August 13, 2018

Periodic Ubuntu Xenial stemcell bump (Aug 14, 2018)

97.5

Release Date: August 08, 2018

Bump Ubuntu Xenial stemcells for “USN-3732-2: Linux kernel (HWE) vulnerability”

97.3

Release Date: July 30, 2018

Periodic Ubuntu Xenial stemcell bump (July 31, 2018)

87.x

This section includes release notes for the 87 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

87.4

Release Date: July 16, 2018

Periodic Ubuntu Xenial stemcell bump (July 16, 2018)

87.3

Release Date: July 11, 2018

Periodic Ubuntu Xenial stemcell bump (July 12, 2018)
87
Release Date: July 02, 2018
Periodic Ubuntu Xenial stemcell bump (July 2, 2018)

81.x
This section includes release notes for the 81 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

81
Release Date: June 19, 2018
- Periodic Ubuntu Xenial stemcell bump (June 18, 2018)

60.x
This section includes release notes for the 60 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

60
Release Date: June 04, 2018
- Periodic Ubuntu Xenial stemcell bump (June 4, 2018)

50.x
This section includes release notes for the 50 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

50
Release Date: May 23, 2018
- Light stemcells are available

40.x
This section includes release notes for the 40 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

40
Release Date: May 22, 2018
- First release of Ubuntu Xenial stemcells
- Notable differences from Ubuntu Trusty
  - Includes systemd instead of upstart
  - Includes 4.15 Linux Kernel instead of 4.4
  - Uses chronyd to sync time (runs as a daemon) instead of ntpdate
  - Does not include NFS utils by default

Warning: Do not downgrade instances from Ubuntu Xenial to Ubuntu Trusty stemcells as it may corrupt persistent disk content since Trusty stemcells may decide to use sfdisk partitioner instead of parted partitioner selected by Xenial stemcells.
7.x
This section includes release notes for the 7 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

7
Release Date: April 08, 2019
Periodic stemcell bump (Apr 8, 2019)

7
Release Date: February 14, 2019
Periodic CentOS 7 stemcell bump (Feb 14, 2019)

7
Release Date: January 28, 2019
Periodic CentOS 7 stemcell bump (Jan 28, 2019)

7
Release Date: January 16, 2019
Periodic CentOS 7 stemcell bump (Jan 17, 2019)

Trusty Stemcells
The following sections describe each Trusty stemcell release.

3763.x
This section includes release notes for the 3763 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3763.14
Release Date: March 11, 2019
Periodic stemcell bump (Mar 14, 2019)

3763.13
Release Date: February 25, 2019
Periodic stemcell bump (Mar 08, 2019)

3586.x
This section includes release notes for the 3586 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).
3586.93
Available in Pivotal Network
Release Date: March 25, 2019
Periodic stemcell bump (Mar 26, 2019)

3586.91
Available in Pivotal Network
Release Date: March 12, 2019
Periodic stemcell bump (Mar 15, 2019)

3586.86
Available in Pivotal Network
Release Date: February 25, 2019
Periodic stemcell bump (Mar 06, 2019)

3586.79
Available in Pivotal Network
Release Date: February 12, 2019
Periodic Ubuntu Trusty stemcell bump (Feb 14, 2019)

3586.71
Available in Pivotal Network
Release Date: January 28, 2019
Periodic Ubuntu Trusty stemcell bump (Jan 28, 2019)

3586.70
Available in Pivotal Network
Release Date: January 23, 2019
Addresses “USN-3866-1: Ghostscript vulnerability” and “USN-3863-1: APT vulnerability”

3586.67
Available in Pivotal Network
Release Date: January 14, 2019
Periodic Ubuntu Trusty stemcell bump (Jan 15, 2019)

3586.65
Available in Pivotal Network

Release Date: December 21, 2018
Periodic Ubuntu Trusty stemcell bump (Dec 21, 2018)

3586.63

Release Date: December 19, 2018
Periodic Ubuntu Trusty stemcell bump (Dec 19, 2018)

3586.60

Available in Pivotal Network

Release Date: December 03, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Dec 05, 2018)

3586.57

Available in Pivotal Network

Release Date: November 19, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Nov 19, 2018)

3586.56

Available in Pivotal Network

Release Date: November 15, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Nov 15, 2018)

3586.54

Available in Pivotal Network

Release Date: November 05, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Nov 05, 2018)

3586.52

Available in Pivotal Network

Release Date: October 22, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Oct 23, 2018)

3586.48

Release Date: October 08, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Oct 11, 2018)
3586.46
Available in Pivotal Network
Release Date: October 02, 2018
Addresses “USN-3776-2: Linux kernel (Xenial HWE) vulnerabilities” (Oct 02, 2018)

3586.43
Available in Pivotal Network
Release Date: September 24, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Sep 25, 2018)

3586.42
Available in Pivotal Network
Release Date: September 10, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Sep 11, 2018)

3586.40
Available in Pivotal Network
Release Date: August 27, 2018
Bump Ubuntu Trusty stemcells for “USN-3756-1: Intel Microcode vulnerabilities”

Known Issue
- On GCP, writing moderate amounts of data to a persistent disk and then migrating the disk will fail with:
  Error: Timed out sending ‘update_settings’. See #159511884

3586.36
Available in Pivotal Network
Release Date: August 14, 2018
Bump Ubuntu Trusty stemcells for “USN-3741-2: Linux kernel (Xenial HWE) vulnerabilities”

3586.35
Release Date: August 13, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Aug 14, 2018)

3586.27
Available in Pivotal Network
Release Date: July 30, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (July 31, 2018)

3586.26
- Available in Pivotal Network
- Release Date: July 16, 2018

Periodic Ubuntu Trusty stemcell bump (July 16, 2018)

3586.25
- Available in Pivotal Network
- Release Date: July 02, 2018
  - Periodic Ubuntu Trusty stemcell bump (July 2, 2018)

3586.24
- Available in Pivotal Network
- Release Date: June 18, 2018
  - Periodic Ubuntu Trusty stemcell bump (June 18, 2018)

3586.23
- Release Date: June 13, 2018
  - We are continuing to investigate GCP stemcell compatibility issue from earlier version, but we did roll back BOSH Agent to an earlier version that seems to not trigger this problem
    - Note: This build does not include fixes to recently published CVE-2018-3665

3586.18
- Release Date: June 04, 2018
  - WARNING: We are currently investigating unresponsive agent issues when using the Google Cloud Platform version of this stemcell. In the meantime, please use 3586.16 when deploying to GCP.
  - Periodic Ubuntu Trusty stemcell bump (June 4, 2018)

3586.16
- Available in Pivotal Network
- Release Date: May 24, 2018
  - Bump Ubuntu Trusty stemcells for “USN-3654-2: Linux kernel (Xenial HWE) vulnerabilities”

TLS for Internal Blobstore Supported

For Ops Manager v2.2 and later, you can enable TLS for your internal blobstore. Make sure you configured all tiles with a stemcell v3586 or later before enabling TLS for your internal blobstore.
For more information, see TLS for Internal Blobstore Supported in the Ops Manager release notes.

3586.8
Release Date: May 21, 2018
- Periodic Ubuntu Trusty stemcell bump (May 21, 2018)

3586.7
Release Date: May 09, 2018
- Bump Ubuntu Trusty stemcells for “USN-3641-1: Linux kernel vulnerabilities”

3586.5
Release Date: May 08, 2018
- Bump s3cli to include AliCloud support
- Bump bosh-agent
  - Support network aliases (used by Softlayer CPI)
  - Support static routes for networks (used by Softlayer CPI)
  - Support ISCSI for persistent disks (used by Softlayer CPI)
  - Use parted when GPT partitions are detected
  - Refactor retryable strategy usages

3541.x
This section includes release notes for the 3541 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3541.65
✔ Available in Pivotal Network
Release Date: December 07, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Dec 07, 2018)

3541.64
✔ Available in Pivotal Network
Release Date: December 03, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Dec 05, 2018)

3541.61
✔ Available in Pivotal Network
Release Date: November 19, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Nov 19, 2018)
3541.60
Available in Pivotal Network
Release Date: November 15, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Nov 15, 2018)

3541.57
Available in Pivotal Network
Release Date: October 22, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Oct 23, 2018)

3541.54
Release Date: October 08, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Oct 11, 2018)

3541.52
Available in Pivotal Network
Release Date: October 02, 2018
Addresses “USN-3776-2: Linux kernel (Xenial HWE) vulnerabilities” (Oct 02, 2018)

3541.49
Available in Pivotal Network
Release Date: September 24, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Sep 25, 2018)

3541.48
Available in Pivotal Network
Release Date: September 10, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Sep 11, 2018)

3541.46
Available in Pivotal Network
Release Date: August 27, 2018
Bump Ubuntu Trusty stemcells for "USN-3756-1: Intel Microcode vulnerabilities"

3541.44
Available in Pivotal Network
Release Date: August 14, 2018
Bump Ubuntu Trusty stemcells for "USN-3741-2: Linux kernel (Xenial HWE) vulnerabilities"

3541.43

Release Date: August 13, 2018
Periodic Ubuntu Trusty/CentOS stemcell bump (Aug 14, 2018)

3541.37
Available in Pivotal Network

Release Date: July 30, 2018
Periodic Ubuntu Trusty stemcell bump (July 31, 2018)

3541.36
Available in Pivotal Network

Release Date: July 16, 2018
Periodic Ubuntu Trusty stemcell bump (July 16, 2018)

3541.35
Available in Pivotal Network

Release Date: July 02, 2018
- Periodic Ubuntu Trusty stemcell bump (July 2, 2018)

3541.34
Available in Pivotal Network

Release Date: June 18, 2018
- Periodic Ubuntu Trusty stemcell bump (June 18, 2018)

3541.31

Release Date: June 04, 2018
- Periodic Ubuntu Trusty stemcell bump (June 4, 2018)

3541.25
Available in Pivotal Network

Release Date: May 09, 2018
- Bump Ubuntu Trusty stemcells for "USN-3641-1: Linux kernel vulnerabilities"
3541.30
Available in Pivotal Network
Release Date: May 23, 2018
• Bump Ubuntu Trusty stemcells for “USN-3654-2: Linux kernel (Xenial HWE) vulnerabilities”

3541.26
Release Date: May 21, 2018
• Periodic Ubuntu Trusty stemcell bump (May 21, 2018)

3541.24
Available in Pivotal Network
Release Date: May 07, 2018
• Ubuntu Trusty stemcells periodic update (May 7, 2018)

3541.12
Available in Pivotal Network
Release Date: April 06, 2018
• Bump Ubuntu Trusty stemcells for USN-3619-2: Linux kernel (Xenial HWE) vulnerabilities

3541.10
Available in Pivotal Network
Release Date: March 26, 2018
• Periodic Ubuntu and CentOS stemcell bump (March 26/27, 2018)

3541.9
Available in Pivotal Network
Release Date: March 12, 2018
• Periodic Ubuntu and CentOS stemcell bump (March 12, 2018)

3541.8
Available in Pivotal Network
Release Date: March 08, 2018
• Bump bosh-agent to 2.67.1
  • Agent will now respect previously set permissions and owner on sys/run, sys/log and data job directories
  • This should fix stemcell compatibility with Diego/Garden if Agent restarts
  • If you were using 3541.x stemcell for any of your deployments, it’s recommended to update your deployments to this version before updating Director since that would cause Agent restart
3541.5
Release Date: February 22, 2018

- Bump Ubuntu Trusty stemcells for USN-3582-2: Linux kernel (Xenial HWE) vulnerabilities

3541.4
Release Date: February 14, 2018

- Rolled back custom umask configuration as we found out it was different in some cases (depends on how processes were started)
  - Hardening of /var/vcap/jobs/* is still applied by the agent

3541.2
Release Date: February 08, 2018

- [breaking] Set default umask to 077 and further harden several /var/vcap/* directories
  - Note that you may have to change your release to adapt to this change
- [breaking] Renamed /var/vcap/bosh/bin/ntpdate to /var/vcap/bosh/bin/sync-time
- [breaking] Stop forwarding SSH events to bosh-agent
  - Agent no longer receives and forwards such events to HM. This should remove a lot of noisy generated by releases that expect a lot of SSH sessions (eg Gitlab). This information will continue to be available in logs forwarded to remote destinations (and locally /var/log/auth.log).
- Fixes env.bosh.swap_size: 0 to work on more clouds (including GCP)

Misc
- Order stemcell tarballs so that upload-stemcell command can execute faster
- Generate packages.txt within stemcell tarball that includes list of installed packages (previously known under different name)

3469.x
This section includes release notes for the 3469 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3469.1
Release Date: February 27, 2018

- Stemcell produced for testing rsyslog bump to the latest version
  - Unless testing rsyslog, use 3468.x or 3541.x stemcell lines

3468.x
This section includes release notes for the 3468 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3468.78
- Available in Pivotal Network

Release Date: October 22, 2018

Periodic Ubuntu Trusty stemcell bump (Oct 23, 2018)
3468.75

Release Date: October 08, 2018
Periodic Ubuntu Trusty stemcell bump (Oct 11, 2018)

3468.73

Available in Pivotal Network
Release Date: October 02, 2018
Addresses “USN-3776-2: Linux kernel (Xenial HWE) vulnerabilities” (Oct 02, 2018)

3468.71

Available in Pivotal Network
Release Date: September 24, 2018
Periodic Ubuntu Trusty stemcell bump (Sep 25, 2018)

3468.69

Available in Pivotal Network
Release Date: September 10, 2018
Periodic Ubuntu Trusty stemcell bump (Sep 11, 2018)

3468.67

Available in Pivotal Network
Release Date: August 27, 2018
Bump Ubuntu Trusty stemcells for “USN-3756-1: Intel Microcode vulnerabilities”

3468.64

Available in Pivotal Network
Release Date: August 14, 2018
Bump Ubuntu Trusty stemcells for “USN-3741-2: Linux kernel (Xenial HWE) vulnerabilities”

3468.63

Available in Pivotal Network
Release Date: August 13, 2018
Periodic Ubuntu Trusty stemcell bump (Aug 14, 2018)

3468.55

Available in Pivotal Network
Release Date: July 30, 2018
Periodic Ubuntu Trusty stemcell bump (July 31, 2018)

3468.54
Available in Pivotal Network

Release Date: July 16, 2018
Periodic Ubuntu Trusty stemcell bump (July 16, 2018)

3468.53

3468.51
Available in Pivotal Network

Release Date: June 18, 2018
  • Periodic Ubuntu Trusty stemcell bump (June 18, 2018)

3468.47
Release Date: June 04, 2018
  • Periodic Ubuntu Trusty stemcell bump (June 4, 2018)

3468.42
Available in Pivotal Network

Release Date: May 09, 2018
  • Bump Ubuntu Trusty stemcells for "USN-3641-1: Linux kernel vulnerabilities"

3468.46
Available in Pivotal Network

Release Date: May 23, 2018
  • Bump Ubuntu Trusty stemcells for "USN-3654-2: Linux kernel (Xenial HWE) vulnerabilities"

3468.44
Release Date: May 21, 2018
  • Periodic Ubuntu Trusty stemcell bump (May 21, 2018)

3468.41
<table>
<thead>
<tr>
<th>Version</th>
<th>Available in Pivotal Network</th>
<th>Release Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3468.30</td>
<td></td>
<td>May 07, 2018</td>
<td>Ubuntu Trusty stemcells periodic update (May 7, 2018)</td>
</tr>
<tr>
<td>3468.28</td>
<td></td>
<td>April 06, 2018</td>
<td>Bump Ubuntu Trusty stemcells for USN-3619-2: Linux kernel (Xenial HWE) vulnerabilities</td>
</tr>
<tr>
<td>3468.27</td>
<td></td>
<td>March 26, 2018</td>
<td>Periodic Ubuntu and CentOS stemcell bump (March 26/27, 2018)</td>
</tr>
<tr>
<td>3468.26</td>
<td></td>
<td>March 01, 2018</td>
<td>Includes updated ixgbevf 4.3.4</td>
</tr>
<tr>
<td>3468.25</td>
<td></td>
<td>February 22, 2018</td>
<td>Bump Ubuntu Trusty stemcells for USN-3582-2: Linux kernel (Xenial HWE) vulnerabilities</td>
</tr>
<tr>
<td>3468.22</td>
<td></td>
<td>February 05, 2018</td>
<td>[Feb 5] Periodic stemcell bump</td>
</tr>
<tr>
<td>3468.21</td>
<td></td>
<td>January 23, 2018</td>
<td>No functional change from 3468.20, except version number</td>
</tr>
</tbody>
</table>
3468.20
Available in Pivotal Network
Release Date: January 23, 2018

- Bump Ubuntu Trusty stemcells for USN-3540-2: Linux kernel (Xenial HWE) vulnerabilities (This flaw is known as Spectre.)

3468.19
Available in Pivotal Network
Release Date: January 17, 2018

- Bump Ubuntu Trusty stemcells for USN-3534-1: GNU C Library vulnerabilities

3468.17
Available in Pivotal Network
Release Date: January 10, 2018

- Bump Ubuntu Trusty stemcells for USN-3522-4: Linux kernel (Xenial HWE) regression
- Configure vSphere stemcells to use VM hardware version 9 (requires ESXi 5.1) to be compatible with https://www.vmware.com/us/security/advisories/VMSA-2018-0004.html

3468.16
Available in Pivotal Network
Release Date: January 10, 2018

- Bump Ubuntu Trusty stemcells for USN-3522-2: Linux (Xenial HWE) vulnerability (This flaw is known as Meltdown.)

3468.15
Release Date: December 15, 2017

- Bump Ubuntu Trusty stemcells for USN-3509-4: Linux kernel (Xenial HWE) regression

3468.13
Available in Pivotal Network
Release Date: December 06, 2017

- Bump Ubuntu Trusty stemcell USN-3509-2: Linux kernel (Xenial HWE) vulnerabilities

3468.12
Release Date: December 06, 2017

- Bump Ubuntu Trusty stemcells for USN-3505-1: Linux firmware vulnerabilities

3468.11
Release Date: November 21, 2017
- Periodic Ubuntu stemcells update
- Includes Agent changes to support IPv6 on vSphere (manual networking)

3468.5
Release Date: October 26, 2017
- Configure `chmod` to have sticky bit set

3468.1
Release Date: October 23, 2017
- Periodic stemcell bump

3468
Release Date: October 05, 2017
- Removed password authentication for Warden stemcells
- Various minor tweaks that were already backported to older lines

Upcoming features on this stemcell line:
- IPv6 support for vSphere

3445.x
This section includes release notes for the 3445 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3445.76
- Available in Pivotal Network
Release Date: October 22, 2018
Periodic Ubuntu Trusty stemcell bump (Oct 23, 2018)

3445.73
Release Date: October 08, 2018
Periodic Ubuntu Trusty stemcell bump (Oct 11, 2018)

3445.71
- Available in Pivotal Network
Release Date: October 02, 2018
Addresses "USN-3776-2: Linux kernel (Xenial HWE) vulnerabilities" (Oct 02, 2018)

3445.68
- Available in Pivotal Network
Release Date: September 24, 2018

Periodic Ubuntu Trusty stemcell bump (Sep 25, 2018)

3445.67
✓ Available in Pivotal Network

Release Date: September 10, 2018

Periodic Ubuntu Trusty stemcell bump (Sep 10, 2018)

3445.66
✓ Available in Pivotal Network

Release Date: August 27, 2018

Bump Ubuntu Trusty stemcells for “USN-3756-1: Intel Microcode vulnerabilities”

3445.64
✓ Available in Pivotal Network

Release Date: August 14, 2018

Bump Ubuntu Trusty stemcells for “USN-3741-2: Linux kernel (Xenial HWE) vulnerabilities”

3445.63

Release Date: August 13, 2018

Periodic Ubuntu Trusty stemcell bump (Aug 14, 2018)

3445.55
✓ Available in Pivotal Network

Release Date: July 30, 2018

Periodic Ubuntu Trusty stemcell bump (July 31, 2018)

3445.54
✓ Available in Pivotal Network

Release Date: July 16, 2018

Periodic Ubuntu Trusty stemcell bump (July 16, 2018)

3445.53
✓ Available in Pivotal Network

Release Date: July 02, 2018

• Periodic Ubuntu Trusty stemcell bump (July 2, 2018)
3445.51
Available in Pivotal Network
Release Date: June 18, 2018
- Periodic Ubuntu Trusty stemcell bump (June 18, 2018)

3445.49
Release Date: June 04, 2018
- Periodic Ubuntu Trusty stemcell bump (June 4, 2018)

3445.48
Available in Pivotal Network
Release Date: May 23, 2018
- Bump Ubuntu Trusty stemcells for "USN-3654-2: Linux kernel (Xenial HWE) vulnerabilities"

3445.42
Available in Pivotal Network
Release Date: May 01, 2018
- Ubuntu Trusty stemcells periodic update (Apr 30, 2018)

3445.46
Release Date: May 21, 2018
- Periodic Ubuntu Trusty stemcell bump (May 21, 2018)

3445.45
Available in Pivotal Network
Release Date: May 09, 2018
- Bump Ubuntu Trusty stemcells for "USN-3641-1: Linux kernel vulnerabilities"

3445.44
Available in Pivotal Network
Release Date: May 07, 2018
- Ubuntu Trusty stemcells periodic update (May 7, 2018)

3445.32
Available in Pivotal Network
Release Date: April 06, 2018
• Bump Ubuntu Trusty stemcells for USN-3619-2: Linux kernel (Xenial HWE) vulnerabilities

3445.30
✓ Available in Pivotal Network

Release Date: March 26, 2018

• Periodic Ubuntu Trusty stemcell bump (March 26/27, 2018)

3445.29
✓ Available in Pivotal Network

Release Date: March 12, 2018

• Periodic Ubuntu Trusty stemcell bump (March 12, 2018)

3445.28
✓ Available in Pivotal Network

Release Date: February 22, 2018

• Bump Ubuntu Trusty stemcells for USN-3582-2: Linux kernel (Xenial HWE) vulnerabilities

3445.25

Release Date: February 05, 2018

• [Feb 5] Periodic stemcell bump

3445.24
✓ Available in Pivotal Network

Release Date: January 23, 2018

• Bump Ubuntu Trusty stemcells for USN-3540-2: Linux kernel (Xenial HWE) vulnerabilities (This flaw is known as Spectre.)

3445.23
✓ Available in Pivotal Network

Release Date: January 17, 2018

• Bump Ubuntu Trusty stemcells for USN-3534-1: GNU C Library vulnerabilities

3445.22
✓ Available in Pivotal Network

Release Date: January 10, 2018

• Bump Ubuntu Trusty stemcells for USN-3522-4: Linux kernel (Xenial HWE) regression

• Configure vSphere stemcells to use VM hardware version 9 (requires ESXi 5.1) to be compatible with https://www.vmware.com/us/security/advisories/VMSA-2018-0004.html
3445.21
Available in Pivotal Network
Release Date: January 10, 2018
• Bump Ubuntu Trusty stemcells for USN-3522-2: Linux (Xenial HWE) vulnerability (This flaw is known as Meltdown.)

3445.19
Available in Pivotal Network
Release Date: December 08, 2017
• Bump Ubuntu Trusty stemcell USN-3509-2: Linux kernel (Xenial HWE) vulnerabilities

3445.18
Release Date: December 06, 2017
• Bump Ubuntu Trusty stemcells for USN-3505-1: Linux firmware vulnerabilities

3445.17
Available in Pivotal Network
Release Date: November 21, 2017
• Periodic Ubuntu stemcells update

3445.11
Available in Pivotal Network
Release Date: September 19, 2017
• Bump Ubuntu stemcells for USN-3420-2: Linux kernel (Xenial HWE) vulnerabilities

3445.7
Available in Pivotal Network
Release Date: August 30, 2017
• Logrotate /var/log/wtmp and utmp more aggressively
• Updated BOSH agent to include aggressive 5 minute timeout on NATS connection failure
• Set auditd rules to be mutable by default
  • Please use audited job from os-conf-release to make rules immutable

3445.2
Release Date: August 16, 2017
• Bump Ubuntu stemcells for USN-3392-2: Linux kernel (Xenial HWE) regression
3445

**Release Date:** August 11, 2017

- Bump Ubuntu stemcells for [USN-3385-2: Linux kernel (Xenial HWE) vulnerabilities](https://security.ubuntu.com/usn/usn-3385-2)  
- Include gcscli for native GCS support  
- Bump bosh-agent to include support for colocated errand jobs  
- Fix occasional rsyslog hang on startup (as a workaround for [https://github.com/rsyslog/rsyslog/issues/1188](https://github.com/rsyslog/rsyslog/issues/1188))

3431.x

This section includes release notes for the 3431 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3431.13

**Release Date:** August 03, 2017

- Bump version (no change)

3431.11

**Release Date:** August 03, 2017

- Bump Ubuntu Trusty stemcells for [USN-3378-2: Linux kernel (Xenial HWE) vulnerabilities](https://security.ubuntu.com/usn/usn-3378-2)  
- Fix occasional rsyslog hang on startup  
  - Workaround for [https://github.com/rsyslog/rsyslog/issues/1188](https://github.com/rsyslog/rsyslog/issues/1188)

3431.10

**Release Date:** July 31, 2017

- Periodic Ubuntu stemcells update

3422.x

This section includes release notes for the 3422 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3422.7

**Release Date:** November 22, 2017

- Test stemcell for umask changes (based on 3468.x stemcell line)

3421.x

This section includes release notes for the 3421 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3421.88

**Release Date:** October 08, 2018

Periodic Ubuntu Trusty stemcell bump (Oct 11, 2018)
3421.86
Available in Pivotal Network
Release Date: October 02, 2018
Addresses “USN-3776-2: Linux kernel (Xenial HWE) vulnerabilities” (Oct 02, 2018)

3421.83
Available in Pivotal Network
Release Date: September 24, 2018
Periodic Ubuntu Trusty stemcell bump (Sep 25, 2018)

3421.82
Available in Pivotal Network
Release Date: September 10, 2018
Periodic Ubuntu Trusty stemcell bump (Sep 10, 2018)

3421.81
Available in Pivotal Network
Release Date: August 27, 2018
Bump Ubuntu Trusty stemcells for “USN-3756-1: Intel Microcode vulnerabilities”

3421.79
Available in Pivotal Network
Release Date: August 14, 2018
Bump Ubuntu Trusty stemcells for “USN-3741-2: Linux kernel (Xenial HWE) vulnerabilities”

3421.78
Release Date: August 13, 2018
Periodic Ubuntu Trusty stemcell bump (Aug 14, 2018)

3421.70
Available in Pivotal Network
Release Date: July 30, 2018
Periodic Ubuntu Trusty stemcell bump (July 31, 2018)

3421.69
Available in Pivotal Network
<table>
<thead>
<tr>
<th>Release Date</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 16, 2018</td>
<td>3421.86</td>
<td>Periodic Ubuntu Trusty stemcell bump (July 16, 2018)</td>
</tr>
<tr>
<td>July 02, 2018</td>
<td>3421.66</td>
<td>Periodic Ubuntu Trusty stemcell bump (July 2, 2018)</td>
</tr>
<tr>
<td>June 18, 2018</td>
<td>3421.64</td>
<td>Periodic Ubuntu Trusty stemcell bump (June 18, 2018)</td>
</tr>
<tr>
<td>June 04, 2018</td>
<td>3421.63</td>
<td>Periodic Ubuntu Trusty stemcell bump (June 4, 2018)</td>
</tr>
<tr>
<td>May 23, 2018</td>
<td>3421.56</td>
<td>Bump Ubuntu Trusty stemcells for “USN-3654-2: Linux kernel (Xenial HWE) vulnerabilities”</td>
</tr>
<tr>
<td>May 01, 2018</td>
<td>3421.60</td>
<td>Ubuntu Trusty stemcells periodic update (Apr 30, 2018)</td>
</tr>
<tr>
<td>May 21, 2018</td>
<td>3421.59</td>
<td>Periodic Ubuntu Trusty stemcell bump (May 21, 2018)</td>
</tr>
<tr>
<td>May 09, 2018</td>
<td>3421.58</td>
<td>Bump Ubuntu Trusty stemcells for “USN-3641-1: Linux kernel vulnerabilities”</td>
</tr>
</tbody>
</table>
Available in Pivotal Network

**Release Date:** May 07, 2018

- Ubuntu Trusty stemcells periodic update (May 7, 2018)

3421.46

Available in Pivotal Network

**Release Date:** April 06, 2018

- Bump Ubuntu Trusty stemcells for USN-3619-2: Linux kernel (Xenial HWE) vulnerabilities

3421.44

Available in Pivotal Network

**Release Date:** March 26, 2018

- Periodic Ubuntu Trusty stemcell bump (March 26/27, 2018)

3421.43

Available in Pivotal Network

**Release Date:** March 12, 2018

- Periodic Ubuntu Trusty stemcell bump (March 12, 2018)

3421.42

Available in Pivotal Network

**Release Date:** February 22, 2018

- Bump Ubuntu Trusty stemcells for USN-3582-2: Linux kernel (Xenial HWE) vulnerabilities

3421.39

**Release Date:** February 05, 2018

- [Feb 5] Periodic stemcell bump

3421.38

Available in Pivotal Network

**Release Date:** January 23, 2018

- Bump Ubuntu Trusty stemcells for USN-3540-2: Linux kernel (Xenial HWE) vulnerabilities (This flaw is known as Spectre.)

3421.37

Available in Pivotal Network

**Release Date:** January 17, 2018

© Copyright Pivotal Software Inc, 2013-2019
- Bump Ubuntu Trusty stemcells for USN-3534-1: GNU C Library vulnerabilities

3421.36
Available in Pivotal Network
Release Date: January 10, 2018
- Bump Ubuntu Trusty stemcells for USN-3522-4: Linux kernel (Xenial HWE) regression
- Configure vSphere stemcells to use VM hardware version 9 (requires ESXi 5.1) to be compatible with

3421.35
Available in Pivotal Network
Release Date: January 10, 2018
- Bump Ubuntu Trusty stemcells for USN-3522-2: Linux (Xenial HWE) vulnerability (This flaw is known as Meltdown.)

3421.34
Available in Pivotal Network
Release Date: December 08, 2017
- Bump Ubuntu Trusty stemcell USN-3509-2: Linux kernel (Xenial HWE) vulnerabilities

3421.33
Release Date: December 06, 2017
- Bump Ubuntu Trusty stemcells for USN-3505-1: Linux firmware vulnerabilities

3421.32
Available in Pivotal Network
Release Date: November 21, 2017
- Periodic Ubuntu stemcells update

3421.20
Available in Pivotal Network
Release Date: August 16, 2017
- Bump Ubuntu stemcells for USN-3392-2: Linux kernel (Xenial HWE) regression

3421.19
Available in Pivotal Network
Release Date: August 11, 2017
- Bump Ubuntu stemcells for USN-3385-2: Linux kernel (Xenial HWE) vulnerabilities
3421.18

Available in Pivotal Network

Release Date: August 03, 2017

- Bump Ubuntu Trusty stemcells for [USN-3378-2: Linux kernel (Xenial HWE) vulnerabilities](https://github.com/usn/usn-3378-2)
- Fix occasional rsyslog hang on startup
  - Workaround for [https://github.com/rsyslog/rsyslog/issues/1188](https://github.com/rsyslog/rsyslog/issues/1188)

3421.11

Release Date: June 29, 2017

- Bump Ubuntu stemcells for USN-3344-2: Linux kernel (Xenial HWE) vulnerabilities

3421.9

Available in Pivotal Network

Release Date: June 21, 2017

- Bump Ubuntu stemcells for USN-3334-1: Linux kernel (Xenial HWE) vulnerabilities

3421.6

Release Date: June 09, 2017

- Bump Ubuntu stemcells for USN-3312-2 - Linux kernel vulnerabilities

3421.4

Release Date: June 01, 2017

- Bump CentOS stemcells for CESA-2017:1382 - sudo vulnerability

3421.3

Available in Pivotal Network

Release Date: May 30, 2017

- Bump Ubuntu stemcells for USN-3304-1: Sudo vulnerability

3421

Release Date: May 22, 2017

New:

- Added `env.bosh.remove_static_libraries` [bool] to remove static libraries
  - Useful to enable this option when exporting compiled releases
- Added `env.bosh.ipv6.enable` [bool] to remove ipv6.disable kernel functionality at bootup time

Fixes:

- Fixed sysstat logging
Fixed anacron's RANDOM_DELAY configuration

Bumps:

- Bumped s3cli v0.0.60
  - Updated aws-sdk-go to solve network timeout edge case
- Bumped davcli v0.0.19
  - Use TCP keep alive to solve network timeout edge case
- Bumped bosh-agent v0.0.35
  - Add `-v` to the Agent binary
  - Prepared `sync_dns` action to work with future Director's DNS integration

3363.x

This section includes release notes for the 3363 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3363.65

Available in Pivotal Network

Release Date: June 18, 2018

- Periodic Ubuntu Trusty stemcell bump (July 18, 2018)

3363.64

Release Date: June 04, 2018

- Periodic Ubuntu Trusty stemcell bump (June 4, 2018)

3363.63

Available in Pivotal Network

Release Date: May 23, 2018

- Bump Ubuntu Trusty stemcells for "USN-3654-2: Linux kernel (Xenial HWE) vulnerabilities"

3363.62

Release Date: May 21, 2018

- Periodic Ubuntu Trusty stemcell bump (May 21, 2018)

3363.61

Available in Pivotal Network

Release Date: May 09, 2018

- Bump Ubuntu Trusty stemcells for "USN-3641-1: Linux kernel vulnerabilities"

3363.60
Available in Pivotal Network

Release Date: May 07, 2018

- Ubuntu Trusty stemcells periodic update (May 7, 2018)

3363.53

Release Date: April 06, 2018

- Bump Ubuntu Trusty stemcells for USN-3619-2: Linux kernel (Xenial HWE) vulnerabilities

3363.52

Release Date: March 26, 2018

- Periodic Ubuntu Trusty stemcell bump (March 26/27, 2018)

3363.51

Release Date: March 12, 2018

- Periodic Ubuntu Trusty stemcell bump (March 12, 2018)

3363.50

Release Date: February 22, 2018

- Bump Ubuntu Trusty stemcells for USN-3582-2: Linux kernel (Xenial HWE) vulnerabilities

3363.49

Release Date: February 05, 2018

- [Feb 5] Periodic stemcell bump

3363.48

Release Date: January 23, 2018

- Bump Ubuntu Trusty stemcells for USN-3540-2: Linux kernel (Xenial HWE) vulnerabilities (This flaw is known as Spectre.)

3363.47

Release Date: January 17, 2018
- Bump Ubuntu Trusty stemcells for USN-3534-1: GNU C Library vulnerabilities

3363.46
Available in Pivotal Network
Release Date: January 10, 2018
- Bump Ubuntu Trusty stemcells for USN-3522-4: Linux kernel (Xenial HWE) regression
- Configure vSphere stemcells to use VM hardware version 9 (requires ESXi 5.1) to be compatible with https://www.vmware.com/us/security/advisories/VMSA-2018-0004.html

3363.45
Available in Pivotal Network
Release Date: January 10, 2018
- Bump Ubuntu Trusty stemcells for USN-3522-2: Linux (Xenial HWE) vulnerability (This flaw is known as Meltdown.)

3363.44
Available in Pivotal Network
Release Date: December 08, 2017
- Bump Ubuntu Trusty stemcell USN-3509-2: Linux kernel (Xenial HWE) vulnerabilities

3363.43
Release Date: December 06, 2017
- Bump Ubuntu Trusty stemcells for USN-3505-1: Linux firmware vulnerabilities

3363.42
Available in Pivotal Network
Release Date: November 21, 2017
- Periodic Ubuntu stemcells update

3363.24
Available in Pivotal Network
Release Date: May 17, 2017
- Periodic Ubuntu stemcells update

3363.22
Release Date: May 11, 2017
- Periodic Ubuntu stemcells update
- Run cron in BOSH Lite stemcells so that logrotation is performed

© Copyright Pivotal Software Inc, 2013-2019
3363.20

Available in Pivotal Network

Release Date: April 25, 2017

- Bump Ubuntu stemcells for USN-3265-2: Linux kernel (Xenial HWE) vulnerabilities

3363.19

Release Date: April 17, 2017

- Periodic bump for CentOS stemcells to include CESA-2017:0933
- Disable IPv6 through /proc/cmdline to eliminate possibility of listening on tcp6/udp6

3363.15

Available in Pivotal Network

Release Date: April 05, 2017

- Bump Ubuntu stemcells for USN-3256-2: Linux kernel (HWE) vulnerability

Misc:
- Made AWS AMI backing snapshot public to support encryption of boot disks

3363.14

Available in Pivotal Network

Release Date: March 30, 2017

- Bump Ubuntu stemcells for USN-3249-2: Linux kernel (Xenial HWE) vulnerability

3363.10

Available in Pivotal Network

Release Date: March 08, 2017

- Bumps Ubuntu stemcells for USN-3220-2: Linux kernel (Xenial HWE) vulnerability

3363.9

Release Date: February 22, 2017

Changes:
- Bumps Ubuntu stemcells for USN-3208-2: Linux kernel (Xenial HWE) vulnerabilities
- Fixes excessive “out of memory” errors in kernel
- Fixes regression to rsyslog by locking it down again to rsyslog 8.22.0

Agent:
- Fixes Azure stemcell persistent disk formatting
- Fixes Warden stemcells SSH access

3363.1

Release Date: February 15, 2017

Reported Problems:
- DO NOT USE azure stemcell as it may cause data loss.
- rsyslog version updated to 8.24.0, regressing on issue #1537
- Out of memory errors still exists in Kernel 4.4.0.62
- will be fixed around Feb 20.
Changes: - Fixes double -hvm- suffix problem for AWS Light stemcells

3363

Release Date: February 15, 2017

Reported Problems: - DO NOT USE azure stemcell as it may cause data loss. - Out of memory errors still exists in Kernel 4.4.0.62 - will be fixed around Feb 20. - rsyslog version updated to 8.24.0, regressig on issue #1537 - AWS Light stemcell has incorrect name once imported - BOSH SSH does not work on BOSH Lite

Changes: - Add more auditd rules - Fix CentOS initramfs to load necessary kernel modules - Disable boot loader login - Increasing tcp_max_sync_backlog - Disabling any DSA host keys - Add bosh_sshers group and assign it to vcap user - Only allow users in bosh_sshers group to SSH

Agent: - Log Agent API access events in CEF format to syslog (vcap.agent topic) - Allow configuring swap size through env:bohs.swap_size (example: env:bohs.swap_size: 0 ) - Prepare for SHA2 releases - Allow setting fetching to work with base64 encoded user data - Do not delaycompress in logrotate

3312.x

This section includes release notes for the 3312 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3312.51

Available in Pivotal Network

Release Date: January 23, 2018

- Bump Ubuntu Trusty stemcells for USN-3540-2: Linux kernel (Xenial HWE) vulnerabilities (This flaw is known as Spectre.)

3312.50

Available in Pivotal Network

Release Date: January 10, 2018

- Bump Ubuntu Trusty stemcells for USN-3522-4: Linux kernel (Xenial HWE) regression

  Configure vSphere stemcells to use VM hardware version 9 (requires ESXi 5.1) to be compatible with https://www.vmware.com/us/security/advisories/VMSA-2018-0004.html

3312.49

Available in Pivotal Network

Release Date: January 09, 2018

- Bump Ubuntu Trusty stemcells for USN-3522-2: Linux (Xenial HWE) vulnerability (This flaw is known as Meltdown.)

3312.48

Available in Pivotal Network

Release Date: December 08, 2017

- Bump Ubuntu Trusty stemcell USN-3509-2: Linux kernel (Xenial HWE) vulnerabilities

3312.47
Release Date: December 06, 2017

- Bump Ubuntu Trusty stemcells for USN-3505-1: Linux firmware vulnerabilities

3312.46

Available in Pivotal Network

Release Date: November 21, 2017

- Periodic Ubuntu stemcells update

3312.17

Release Date: January 31, 2017

Reported Problems: - Memory leak bug in Kernel 4.4.0-59

Changes: - Periodic stemcell update

3312.15

Release Date: January 12, 2017

Reported Problems: - Memory leak bug in Kernel 4.4.0-59

Changes: - Periodic stemcell update

3312.12

Available in Pivotal Network

Release Date: December 20, 2016

- Lock down rsyslog to 8.22.0 to avoid high memory usage issue https://github.com/cloudfoundry/bosh/issues/1537

3312.9

Available in Pivotal Network

Release Date: December 15, 2016

- Updates Azure Ubuntu stemcells to use parted partitioner to fix slow disk partitioning
  - Related issue: https://github.com/cloudfoundry-incubator/bosh-azure-cpi-release/issues/227

3312.8

Release Date: December 14, 2016

- Bumps Ubuntu stemcells for USN-3156-1: APT vulnerability

3312.7

Release Date: December 06, 2016

- Bump Ubuntu stemcells for USN-3151-2: Linux kernel (Xenial HWE) vulnerability
3312.6
Release Date: December 02, 2016
- Periodic stemcell update

3312.3
Release Date: November 30, 2016
- Periodic stemcell update
  - Includes USN-3134-1 as requested by a community member

3312
Release Date: November 16, 2016
- Properly includes libpam_cracklib.so to avoid errors in /var/log/auth.log

3309.x
This section includes release notes for the 3309 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3309
Release Date: November 10, 2016
- Fixes persistent disk mounting on OpenStack described in Stemcell 3308

3308.x
This section includes release notes for the 3308 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3308
Release Date: November 09, 2016
Reported Problems: - On OpenStack: Mounting persistent disks not working when using config-drive: disk while nova is configured to use a cdrom config-drive due to https://github.com/cloudfoundry/bosh/issues/1503
- Fixes: - Fixes SSH key installation issue introduced in Stemcell 3306

3306.x
This section includes release notes for the 3306 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3306
Release Date: November 08, 2016
Reported Problems - bosh-init doesn’t work with this stemcell on OpenStack and AWS due to https://github.com/cloudfoundry/bosh/issues/1500 - Booting the stemcell image directly in your IaaS (without using BOSH/bosh-init) does no longer provision the ssh key for user vcap, so you need to login differently
Changes - Agent will now wait for monit to complete stop all processes before carrying on - Added google stemcells - Default dmesg_restrict to 1 - Disable
all IPv6 configurations - Reenabled UDF kernel module for Azure - Increase root_maxkeys and maxkeys kernel configurations - Changed default hostname to bosh-stemcell instead of localhost to avoid boot problems on GCP - Lower TCP keepalive configuration by default - Mount /var/log directory to /var/ucap/data/root_log - Restrict Access to the su command - Add pam_cracklib requirements to common-password and password-auth - Enable auditing for processes that start prior to auditd - Set log rotation interval to 15 min in stemcell - Made ownership & permissions for /etc/cron* files more restrictive - Customize shell prompt to show instance name and ID - Removed floppy drives from vSphere stemcells - Removed bosh-micro assets hence making bosh-micro unsupported

Misc: - Stemcells are now built through Concourse via https://main.bosh-ci.cf-app.com/teams/main/pipelines/bosh:stemcells

3263.x
This section includes release notes for the 3263 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3263.10
Release Date: November 03, 2016
- Updates CentOS kernel to the latest version for "Dirty COW"
  - Ubuntu stemcells were updated in previous versions at the time of Ubuntu USN updates
  - Includes fix to the bosh-agent to better support 1TB+ disk partitioning

3263.8
☑ Available in Pivotal Network
Release Date: October 21, 2016
- Bump Ubuntu stemcells for USN-3106-2: Linux kernel (Xenial HWE) vulnerability
- Includes a fix to the bosh-agent to work more reliably with 2TB+ persistent disks

3263.7
☑ Available in Pivotal Network
Release Date: October 12, 2016
- Bump Ubuntu stemcells for USN-3099-2: Linux kernel (Xenial HWE) vulnerabilities

3263.5
Release Date: September 30, 2016
- Periodic bump
  - Delay start of rsyslogd using systemd on CentOS

3263.4
Release Date: September 28, 2016
- google-kvm: improve the google- daemon configurations
  - fixes ssh handshake failed errors on boot
3263.3

Available in Pivotal Network

Release Date: September 26, 2016

- Bumps Ubuntu stemcells for USN-3087-2 (OpenSSL regression)

3263

Release Date: September 19, 2016

- Bumps Ubuntu to Linux kernel to 4.4

Based on 3262 stemcells. Note: OpenStack stemcells series 3263 is broken due to https://github.com/cloudfoundry/bosh-agent/issues/98 and should not be used.

3262.x

This section includes release notes for the 3262 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3262.21

Available in Pivotal Network

Release Date: October 13, 2016

- Bump Ubuntu stemcells for USN-3099-2: Linux kernel (Xenial HWE) vulnerabilities

3262.19

Available in Pivotal Network

Release Date: September 28, 2016

- google-kvm: improve the google-* daemon configurations
  - fixes ssh handshake failed errors on boot

3262.16

Available in Pivotal Network

Release Date: September 26, 2016

- Bumps Ubuntu stemcells for USN-3087-2 (OpenSSL regression)

3262.15

Release Date: September 23, 2016

- Bumps Ubuntu stemcells for USN-3087-1: OpenSSL vulnerabilities

3233.x

This section includes release notes for the 3233 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).
USN-3522-2 Addresses Meltdown Vulnerabilities

Meltdown exploits critical vulnerabilities in modern processors. For more information about Meltdown, see the Meltdown and Spectre Attacks blog post. USN-3522-2 addresses the critical vulnerability in Ubuntu associated with Meltdown.

This update may include degradations to performance if your VM’s CPU and memory usage are currently at near-capacity levels. Prior to upgrading to this stemcell, monitor your PCF VM’s current CPU and memory usage and scale those components if necessary. If any of your VMs are currently operating at 60% or above, Pivotal recommends scaling that VM. For more information about the performance impact of Meltdown-related stemcell patches on PCF components and guidance on scaling, see this KB article.

For more information about monitoring and scaling PCF, see the Monitoring PCF VMs from Ops Manager, Key Capacity Scaling Indicators, and Scaling PAS topics. Performance degradation is likely to vary by workload type, IaaS, and other factors. Pivotal recommends testing your deployment thoroughly after upgrading to this stemcell.

3232.x

This section includes release notes for the 3232 line of Linux stemcells used with Pivotal Cloud Foundry (PCF).

3232.21

Available in Pivotal Network

Release Date: September 26, 2016

• Bumps Ubuntu stemcells for USN-3087-2 (OpenSSL regression)

This topic includes release notes for Windows stemcells used with Pivotal Application Service for Windows (PASW) v2.5.

The stemcell is based on Windows Server, version 2019.

To download a stemcell, see Stemcells for PCF (Windows) on Pivotal Network.

2019.3

Release Date: April 25, 2019

Features

- This is the first 2019 stemcell.
- Includes Microsoft Security Updates Patch Tuesday April 2019
Tiles Using Xenial Stemcells in PCF

This table provides a quick reference of PCF product versions that are now using Xenial stemcells.

Products Using Xenial Stemcells

If a product is not yet listed, the product still uses Trusty stemcells.

<table>
<thead>
<tr>
<th>Product/Component</th>
<th>Starting Version</th>
<th>Release Notes Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCF Operations Manager</td>
<td>v2.3</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Pivotal Application Service (PAS)</td>
<td>v2.3</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Pivotal Container Service (PKS)</td>
<td>v1.2</td>
<td>Release Notes</td>
</tr>
<tr>
<td>PCF Event Alerts</td>
<td>v1.2.3</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Metrics Forwarder for PCF</td>
<td>v1.11.3</td>
<td>Release Notes</td>
</tr>
<tr>
<td>MySQL for PCF</td>
<td>v2.4</td>
<td>Release Notes</td>
</tr>
<tr>
<td>RabbitMQ for PCF</td>
<td>v1.14.1</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Redis for PCF</td>
<td>v1.14.0</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Scheduler for PCF</td>
<td>v1.2.3</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Single Sign-On (SSO) for PCF</td>
<td>v1.7.1</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Spring Cloud Data Flow</td>
<td>v1.2.0</td>
<td>Release Notes</td>
</tr>
<tr>
<td>Spring Cloud Services for PCF</td>
<td>v2.0.2</td>
<td>Release Notes</td>
</tr>
</tbody>
</table>

Xenial Stemcell Support in Ops Manager

You can import and use Xenial stemcell-based tiles in the following versions of Ops Manager:

- v2.1.15 and later
- v2.2.2 and later
- v2.3.x and later

For more information about the introduction of Xenial stemcells to your PCF deployment, see Updates for Xenial Stemcell Support in the Pivotal Cloud Foundry v2.3 Breaking Changes Release Notes.
Architecture and Installation Overview

For PCF Architects and Operators

This guide shows you how to design a Pivotal Cloud Foundry (PCF) platform and install it on an IaaS. If you are doing this, you have one or both of the following roles:

- **Architects** design a PCF platform. They know the IaaS that they will deploy it to, and what other relevant resources they have. In their design, they consider needs for the platform’s capacity, availability, security, geography, budget, and other factors. If they do not install PCF themselves, they provide the architectural specifications to whoever does.

- **Operators** run a PCF platform, keep it up-to-date, monitor its health and performance, and fix any problems. They may also install the platform, or perform “Day 2” configurations that expand its functionality and integrate it with external systems.

This guide helps people in both roles create a PCF platform that does what they want it to. The contents of this guide follow the phases of a typical PCF planning and installation effort.

Note: Elastic Runtime has been renamed Pivotal Application Service.

Planning and Installation Overview

PCF is a suite of products that runs on multiple IaaSes. Planning and installing PCF means building layers from the bottom up, starting with the details of your IaaS and ending with “Day 2” configurations that you perform on a installed and running PCF deployment.

Here’s the typical PCF planning and installation process:

1. **Plan Your PCF Architecture**
   - Review the Requirements for your IaaS (AWS, Azure, GCP, OpenStack, vSphere).
   - Under Reference Architectures, refer to the reference architecture for your IaaS.
   - Assess your platform needs, including capacity, availability, container support, host OS, resource isolation, and geographical distribution. Discuss with your Pivotal contact.

2. **Deploy BOSH and Ops Manager**
   - **BOSH** is an open-source tool that lets you run software systems in the cloud.
     - BOSH and its IaaS-specific Cloud Provider Interfaces (CPIs) are what enable PCF to run on multiple IaaSes.
     - See Deploying with BOSH for a brief description of the BOSH deployment process.
   - **Ops Manager** is a GUI application deployed by BOSH that streamlines deploying subsequent software to the cloud via BOSH.
     - Ops Manager represents PCF products as tiles with multiple configuration panes that let you input or select configuration values needed for the product.
     - Ops Manager generates BOSH manifests containing the user-supplied configuration values, and sends them to the Director.
     - After you install Ops Manager and BOSH, you use Ops Manager to deploy almost all PCF products.
   - Deploying Ops Manager deploys both BOSH and Ops Manager with a single procedure.
     - On AWS, you can deploy Ops Manager manually, or automatically with a CloudFormation template.
     - On Azure, you can deploy Ops Manager manually, or automatically with an Azure Resource Manager (ARM) template. On Azure Government Cloud and Azure Germany you can only deploy Ops Manager manually.

3. **Deploy BOSH Add-ons** (Optional)
   - BOSH add-ons include the IPsec, ClamAV, and File Integrity Monitoring, which enhance PCF platform security and security logging.
   - You deploy these add-ons via BOSH rather than installing them with Ops Manager tiles.

4. **Install Runtimes**
   - **PAS** (Pivotal Application Service) lets developers develop and manage cloud-native apps and software services.
     - PAS is based on the Cloud Foundry Foundation’s open-source Application Runtime (formerly Elastic Runtime) project.
   - **PKS** (Pivotal Container Service) uses BOSH to run and manage Kubernetes container clusters.
PKS is based on the Cloud Foundry Foundation's open-source Container Runtime (formerly Kubo) project.

- **PCF Isolation Segment** lets a single PAS deployment run apps from separate, isolated pools of computing, routing, and logging resources.
  - Operators replicate and configure an Isolation Segment tile for each new resource pool they want to create.
  - You must install PAS before you can install Isolation Segment.
- **PAS for Windows 2012R2** enables PAS to manage Windows Server 2012 cells hosting .NET apps, and can also be replicated to create multiple isolated resource pools.
  - As with Isolation Segment, Operators replicate and configure a PAS for Windows 2012R2 tile for each new resource pool they want to create.
  - You must install PAS before you can install PAS for Windows 2012R2.
- **Small Footprint PAS** is an alternative to PAS that uses far fewer VMs than PAS but has limitations.

5. **Install Services**
   - Install software services for PCF developers to use in their apps.
     - Services include the databases, caches, and message brokers that stateless cloud apps rely on to save information.
     - Installing and managing software services on PCF is an ongoing process, and is covered in the [PCF Operator Guide](#).

6. **Day 2 Configurations**
   - Day 2 configurations set up internal operations and external integrations on a running PCF platform.
     - Examples include front-end configuration, user accounts, logging and monitoring, internal security, and container and stemcell images.

**Guide Contents**

This guide has two parts. The first part explains the PCF planning and installation process, and the second describes the main tools that operators use when installing PCF.

**Planning and Installation**

PCF is a suite of products that runs on multiple IaaSes. Planning and installing PCF means building layers from the bottom up, starting with the details of your IaaS and ending with “Day 2” configurations that you perform on a installed and running PCF deployment.

This guide follows this bottom-up progression:

- **Planning PCF** - Design a PCF platform that runs on your IaaS and fits your needs.
- **Deploying BOSH and Ops Manager** - Build the foundation of PCF.
- **Deploying BOSH Add-Ons** - Use BOSH to extend foundation-level PCF capabilities.
- **Using Ops Manager** - Ops Manager provides a dashboard for installing and configuring PCF products and services.
- **Using the Cloud Foundry Command Line Interface (cf CLI)** - The cf CLI runs PCF operations in the cloud from a shell on your local workstation.
- **Installing Runtimes** - Install the application and container runtime environments that PCF exists for.
- **Day 2 Configurations** - Set up internal operations and external integrations for PCF.
- **Troubleshooting and Diagnostics**

After installing PCF, Operators install the software services that PCF developers use in their apps. These PCF services include the databases, caches, and message brokers that stateless cloud apps rely on to save information.

Installing and managing software services on PCF is an ongoing process, and is covered in the [PCF Operator Guide](#).

**Related Documentation**

The [PCF Operator Guide](#) explains how to maintain a running PCF platform, including monitoring, tuning, troubleshooting, and upgrading.

The [PCF Security Guide](#) explains how PCF security systems work and how to keep your PCF platform secure.

[Getting Started with PCF](#) gives a high-level overview of how PCF works and explains how you can try a simple deployment on your own local
machine.

See Pivotal Cloud Foundry Documentation for all PCF documentation.
Preparing Your Firewall

This topic describes how to configure your firewall for Pivotal Cloud Foundry (PCF) and how to verify that PCF resolves DNS entries behind your firewall.

Configure Your Firewall for PCF

Ops Manager and Pivotal Application Service (PAS) require the following open TCP ports:

- **25555**: Routes from Ops Manager to the BOSH Director.
- **443**: Routes to HAProxy or, if configured, your own load balancer.
- **80**: Routes to HAProxy or, if configured, your own load balancer.
- **8844**: Routes from Ops Manager to BOSH CredHub.
- **8443**: Routes from Ops Manager to BOSH Director UAA.
- **6868**: Routes to the BOSH Agent.
- **2222**: Necessary for using Application SSH. For details, see the Diego Architecture Diagram.
- **25595**: Routes from the Traffic Controller to the BOSH Director, to enable sending BOSH health metrics to the Firehose.
- **UDP port 123**: Must be open if you want to use an external NTP server.

For more information about required ports for additional installed products, see Network Communication Paths.

Example: Configure Firewall with iptables

The following example procedure uses `iptables` commands to configure a firewall.

1. Open `/etc/sysctl.conf`, a file that contains configurations for Linux kernel settings, with the command below:

   ```
   $ sudo vi /etc/sysctl.conf
   ``

2. Add the line `net.ipv4.ip_forward=1` to `/etc/sysctl.conf` and save the file.

3. If you want to remove all existing filtering or Network Address Translation (NAT) rules, run the following commands:

   ```
   $ iptables --flush
   $ iptables --flush -t nat
   ```

4. Add environment variables to use when creating the IP rules:

   ```
   $ export INTERNAL_NETWORK_RANGE=10.0.0.0/8
   $ export GATEWAY_INTERNAL_IP=10.0.0.1
   $ export GATEWAY_EXTERNAL_IP=203.0.113.242
   $ export PIVOTALCF_IP=10.0.0.2
   $ export HA_PROXY_IP=10.0.0.254
   ```

5. Run the following commands to configure IP rules for the specified chains:

   - **FORWARD**:

     ```
     $ iptables -A FORWARD -i eth1 -j ACCEPT
     $ iptables -A FORWARD -o eth1 -j ACCEPT
     ```

   - **POSTROUTING**:
6. Run the following command to save the iptables:

```
$ service iptables save
```

For more information about administering IP tables with `iptables`, refer to the [iptables documentation](https://www.centos.org/docs/6/en/xinetd/howto/). 

---

**Verify PCF Resolves DNS Entries Behind a Firewall**

When you install PCF in an environment that uses a strong firewall, the firewall might block DNS resolution. For example, if you use `xip.io` to test your DNS configuration, the tests will fail without warning if the firewall prevents PAS from accessing `*.xip.io`.

To verify that PAS can correctly resolve DNS entries:

1. SSH into the Pivotal Ops Manager VM. For more information, refer to the [SSH into Ops Manager](https://docs.cloudfoundry.org/guides/advanced_troubleshooting.html#manage-the-sshd-service) section of the Advanced Troubleshooting with the BOSH CLI topic.

2. Run any of the following network administration commands with the IP address of the VM:

   - `nslookup`
   - `dig`
   - `host`
   - The appropriate `traceroute` command for your OS

3. Review the output of the command and fix any blocked routes. If the output displays an error message, review the firewall logs to determine which blocked route or routes you need to clear.

4. Repeat steps 1-3 with the BOSH Director VM and the HAProxy VM.
IaaS Permissions Guidelines

This topic describes practices recommended by Pivotal for creating secure IaaS user roles.

Pivotal Cloud Foundry (PCF) is an automated platform that connects to IaaS providers such as AWS and OpenStack. This connectivity typically requires accounts with appropriate permissions to act on behalf of the operator to access IaaS functionality such as creating virtual machines (VMs), managing networks and storage, and other related services.

Ops Manager and Pivotal Application Service (PAS) can be configured with IaaS users in different ways depending on your IaaS. Other product tiles and services might also use their own IaaS credentials. Refer to the documentation for those product tiles or services to configure them securely.

Least Privileged Users (LPUs)

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role. In the event that someone gains access to credentials by mistake or through malicious intent, LPUs limit the scope of the breach. Pivotal recommends following best practices for the particular IaaS you are deploying.

AWS Guidelines

See the recommendations detailed in the AWS Permissions Guidelines topic.

Azure Guidelines

See the permissions recommendations in Preparing to Deploy Ops Manager on Azure Manually, and use the minimum permissions necessary when creating your service principal.

GCP Guidelines

For GCP, Pivotal recommends using two different accounts with the least privilege.

Use one account with the minimum permissions required to create desired GCP resources in your GCP project, then create a separate service account with the minimum permissions required to deploy PCF components such as Pivotal Ops Manager and PAS. For more information about creating the service account, see Step 1: Set up IAM Service Accounts in Preparing to Deploy Ops Manager on GCP Manually.

OpenStack Guidelines

Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role.

vSphere Guidelines

See the vCenter permissions recommendations in the Installing Pivotal Cloud Foundry on vSphere topic.
Installing PCF in Airgapped Environments

This topic provides an overview of the components and resources needed to install PCF in airgapped environments, including the typical corresponding automation resources.

Offline Components

To run PCF offline, you must obtain resources from the internet and move them into offline components that store and use them. The method you use to move a resource from Pivotal Network or GitHub into an offline environment can vary from setting up a designated proxy to burning a DVD.

The following image displays types of resources and the components you must move them to in your offline environment. It also includes a jumpbox. The jumpbox is a Linux host for running commands such as `bosh`, `uaac`, and `fly`. You could use the Ops Manager VM for this purpose.

The following table provides more detail about resources and the component you must move them to:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pivotal Network products such as tiles, stemcells, BOSH releases, and Ops Manager</td>
<td>S3 and artifact store</td>
</tr>
<tr>
<td>Pipelines and scripts</td>
<td>Git</td>
</tr>
<tr>
<td>Configuration</td>
<td>Git</td>
</tr>
<tr>
<td>Container images</td>
<td>Docker Registry, S3, or artifact store</td>
</tr>
<tr>
<td>Third Party Resources such as NSX-T and OSS BOSH releases</td>
<td>Artifact store</td>
</tr>
<tr>
<td>Backup artifacts</td>
<td>S3</td>
</tr>
</tbody>
</table>

Architectural Patterns

The following sections describe three architectural patterns for running PCF in airgapped environments. The pattern you use depends on how your environment is set up.

Pattern One: Artifact Store with Internet Access

It is common for organizations to deploy an artifact store such as Artifactory or Nexus as a gateway to the Internet. In this configuration, the artifact store is typically configured as a Docker mirror.

This pattern includes an S3 component that is used only for backups of the PCF installation.
The following table provides an example of how you might handle resources in this scenario:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pivotal Network products such as tiles, stemcells, BOSH releases, and Ops Manager</td>
<td>This architectural pattern presents the challenge of getting Pivotal Network resources from the Internet into the artifact store. For example, Artifactory and Nexus cannot interact directly with Pivotal Network. You could place Pivotal Network resources in an artifact store either through a remote Concourse worker with access to the Internet, by downloading them from the Internet and placing them in the artifact store manually, or some other method.</td>
</tr>
<tr>
<td>Pipelines and scripts</td>
<td>Store in Git. Use manual clone and push. Modify pipelines to use <code>registry_mirror</code>.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Store in Git. Use manual clone and push.</td>
</tr>
<tr>
<td>Container images</td>
<td>Store in the artifact store that you configured as a mirror.</td>
</tr>
<tr>
<td>Third Party Resources such as NSX-T and OSS BOSH releases</td>
<td>Store in a generic artifact store repository.</td>
</tr>
<tr>
<td>Backup artifacts</td>
<td>Store in a S3 blobstore.</td>
</tr>
</tbody>
</table>

Pattern Two: Completely Offline

In completely offline environments, you must bring in all resources manually and store them in the available local components.
In this architectural pattern, you must configure pipelines to watch components for changes and apply updates when available. The following table provides an example of how you might handle resources in this scenario:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pivotal Network products such as tiles, stemcells, BOSH releases, and Ops Manager</td>
<td>Manually upload to Nexus.</td>
</tr>
<tr>
<td>Pipelines and scripts</td>
<td>Modify pipelines to use a local Harbor Docker registry. Manually clone an online environment, bring it to the offline environment on DVD, and push it to offline Git.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Store in your offline Git.</td>
</tr>
<tr>
<td>Container Images</td>
<td>Get from the Internet, transfer to USB, and push to offline Harbor.</td>
</tr>
<tr>
<td>Third Party Resources such as NSX-T and OSS BOSH releases</td>
<td>Store in a generic artifact store repository.</td>
</tr>
<tr>
<td>Backup artifacts</td>
<td>Store in a S3 blobstore.</td>
</tr>
</tbody>
</table>

**Pattern Three: TLS Intercepting Proxy**

This pattern allows Internet access, but only through a proxy that decrypts and rewrites TLS certificates. In general, this resembles an online install, however it requires that you add your corporate certificate to all BOSH-deployed VMs.
Some of the challenges in this pattern include the following:

- Any automation that calls to the Internet will fail if it does not have the corporate certificate in its trust store.
- Concourse tasks that do not use resources do not receive updated BOSH root certificates. This means you must configure tasks to ignore TLS errors or update the root certificates as part of the tasks.
Installing Pivotal Cloud Foundry on AWS

This guide describes how to install Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS).

Overview

You can install PCF on AWS with either the Pivotal Application Service (PAS) or Pivotal Container Service (PKS) runtime. There are resource requirements specific to each runtime. Ensure you meet the requirements for your runtime and the requirements specific to AWS before installing PCF on AWS.

Requirements

This section lists the following resource requirements for installing PCF on AWS:

- General PCF resource requirements. See PCF Resource Requirements.
- AWS-specific resource requirements. See AWS Resource Requirements.

PCF Resource Requirements

This section lists PCF resource requirements for installing PCF on AWS. It includes general PCF resource requirements for both the PAS and PKS runtimes.

View one of the following, depending on your PCF runtime:

- PAS-specific PCF resource requirements. See PAS Resource Requirements.
- PKS-specific PCF resource requirements. See PKS Resource Requirements.

PAS Resource Requirements

The following are general resource requirements for deploying and managing a PCF deployment with Ops Manager and PAS:

- PAS requires sufficient IP allocation. The following lists the minimum required IP allocations:
  - One static IP address for either HAProxy or one of your Gorouters
  - One static IP address for each job in the Ops Manager tile. See the Ops Manager Resource Config pane for each tile for a full list.
  - One static IP address for each job listed below:
    - Consul
    - NATS
    - File Storage
    - MySQL Proxy
    - MySQL Server
    - Backup Restore Node
    - HAProxy
    - Router
    - MySQL Monitor
    - Diego Brain
    - TCP Router
  - One IP for each VM instance created by the service.
  - An additional IP address for each compilation worker. Use the following formula to determine the total IPs required:
    \[
    \text{IPs needed} = \text{static IPs} + \text{VM instances} + \text{compilation workers}
    \]
  - Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

- Pivotal recommends using a network without DHCP for deploying PAS VMs.

Note: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.
PKS Resource Requirements

For PKS-specific resource requirements, see AWS Prerequisites and Resource Requirements.

AWS Resource Requirements

The following are AWS-specific resource requirements for installing PCF on AWS with an external database and external file storage:

- Installing PCF on AWS requires a minimum of the following VM instance limits in your AWS account. The number of VMs required depends on the number of tiles and availability zones you plan to deploy. The following VM guidelines apply to the PAS, Small Footprint PAS, and PKS runtimes:

  - **PAS:** At a minimum, a new AWS deployment requires the following VMs for PAS:

    | AWS Requirements | VM Name             | VM Type | Default VM Count | Required or Optional VM |
    |------------------|---------------------|---------|------------------|-------------------------|
    |                  | NATS                | t2.micro| 2                | Required                |
    |                  | File Storage        | t4.large| 1                | Optional                |
    |                  | MySQL Proxy         | t2.micro| 2                | Optional                |
    |                  | MySQL Server        | r4.large| 3                | Optional                |
    |                  | Backup Restore Node | t2.micro| 1                | Optional                |
    |                  | Diego BBS           | t2.micro| 3                | Required                |
    |                  | UAA                 | m4.large| 2                | Required                |
    |                  | Cloud Controller    | m4.large| 2                | Required                |
    |                  | HAProxy             | t2.micro| 0                | Optional                |
    |                  | Router              | t2.micro| 3                | Required                |
    |                  | MySQL Monitor       | t2.micro| 1                | Optional                |
    |                  | Clock Global        | t2.medium| 2               | Required                |
    |                  | Cloud Controller Worker| t2.micro| 2             | Required                |
    |                  | Diego Brain         | t2.small| 3                | Required                |
    |                  | Diego Cell          | r4.xlarge| 3              | Required                |
    |                  | Loggregator Traffic Controller| t2.micro| 2             | Required                |
    |                  | Syslog Adapter      | t2.micro| 3                | Required                |
    |                  | Syslog Scheduler    | t2.micro| 2                | Required                |
    |                  | Doppler Server      | m4.large| 3                | Required                |
    |                  | TCP Router          | t2.micro| 0                | Optional                |
    |                  | CredHub             | r4.large| 2                | Optional                |
    |                  | Istio Router        | r4.large| 0                | Optional                |
    |                  | Istio Control       | r4.large| 0                | Optional                |
    |                  | Route Syncer        | r4.large| 0                | Optional                |
    |                  | Ops Manager         | BOSH Director | m4.large| 1          | Required                |

  - **Small Footprint PAS:** To run Small Footprint PAS, a new AWS deployment requires:

    | AWS Requirements | VM Name             | VM Type | Default VM Count | Minimum HA VM Count | Required or Optional VM |
    |------------------|---------------------|---------|------------------|----------------------|-------------------------|
    |                  | Compute             | r4.xlarge| 1                | 3                    | Required                |
    |                  | Control             | r4.xlarge| 1                | 2                    | Required                |
    |                  | Database            | r4.large| 1                | 3                    | Required                |
    |                  | Router              | t2.micro| 1                | 3                    | Required                |
    |                  | File Storage        | m4.large| 1                | N/A                  | Optional                |
    |                  | Backup Restore Node | t2.micro| 1                | N/A                  | Optional                |

*Note:* If you are deploying a test or sandbox PCF that does not require high availability, then you can scale down the number of VM instances in your deployment. For more information, see Scaling PAS.
PKS: See [AWS Prerequisites and Resource Requirements](#).

The following AWS resources are required for installing PCF on AWS with PAS:

- 3 Elastic Load Balancers (ELBs)
- 1 Relational Database Service. As a minimum, Pivotal recommends using a db.m3.xlarge instance with at least 100 GB of allocated storage.
- 5 S3 Buckets

### Prerequisites

To install PCF on AWS, you must do the following:

- Increase or remove the VM instance limits in your AWS account. Installing PCF requires more than the default 20 concurrent instances. For more information about VM resource requirements, see [Requirements](#).
- Configure your AWS account with the appropriate AWS region. For more information about selecting the correct region for your deployment, see [Region and Availability Zone Concepts](#) in the AWS documentation.
- Install the AWS CLI. Configure the AWS CLI with the user credentials that have admin access to your AWS account. To download the AWS CLI, see [AWS CLI](#).
- Configure an AWS EC2 key pair to use with your PCF deployment. For more information, see [Creating an EC2 Key Pair](#) in the AWS documentation.
- Register a wildcard domain for your PCF installation. For more information, see [SSL/TLS Certificates for Classic Load Balancers](#) in the AWS documentation.
- Create an SSL certificate for your PCF domain. For more information, see the [AWS documentation about SSL certificates](#).

**Note:** To deploy PCF to a production environment, you must obtain a certificate from a certificate authority. Pivotal recommends using a self-signed certificate generated by Ops Manager for development and testing purposes only.

- (PAS-only) Configure sufficient IP allocation. For more information about IP allocation requirements, see [PAS Resource Requirements](#).
- (Optional) (PAS-only) Configure external storage. Pivotal recommends using external storage if possible. For more information about how file storage location affects platform performance and stability during upgrades, see [Configure File Storage](#).
- (Optional) (PAS and Ops Manager-only) Configure external databases. Pivotal recommends using external databases in production deployments for BOSH Director and PAS. An external database must be configured to use the UTC timezone.
- (Optional) (PAS and Ops Manager-only) Configure external user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.

### Install PCF on AWS

You can install PCF on AWS either manually or using Terraform.

To install PCF on AWS, do one of the following:

- Install PCF on AWS manually. See [Installing PCF on AWS Manually](#).
- Install PCF on AWS using Terraform. See [Install PCF on AWS Using Terraform](#).

### Additional Resources

The following are additional resources related to installing PCF on AWS:

- For information about AWS identity and access management, see [What is IAM?](#) in the AWS documentation.
For information about users, groups, and roles in AWS, see Identities (Users, Groups, and Roles) in the AWS documentation.

For best practices for managing IaaS users and permissions, see Temporary Security Credentials in the AWS documentation.

For recommendations on how to create and scope AWS accounts for PCF, see AWS Permissions Guidelines.

For more information about certificate requirements for installing PCF, see Certificate Requirements.
**AWS Permissions Guidelines**

Pivotal recommends that you minimize the use of master account credentials by creating an IAM role and instance profile with the minimum required EC2, VPC, and EBS credentials.

In addition, Pivotal recommends that you follow AWS account security best practices such as disabling root keys, using multi-factor authentication on the root account, and using CloudTrail for auditing API actions.

For more Amazon-specific best practices, refer to the following Amazon documentation:

- IAM Roles Best Practices
- AWS Security Best Practices Whitepaper
- AWS Well-Architected Framework
Installing PCF on AWS Manually

Page last updated:

This topic describes how to manually configure the Amazon Web Services (AWS) components that you need to run Pivotal Cloud Foundry (PCF) on AWS.

To deploy PCF on AWS, you must perform the procedures in this topic to create objects in the AWS Management Console that PCF requires.

To view the list of AWS objects created by the procedures in this topic, see the Required AWS Objects section.

After completing the procedures in this topic, proceed to Configuring BOSH Director on AWS to continue deploying PCF.

Step 1: File a Ticket

Log in to the AWS Management Console, and file a ticket with Amazon to ensure that your account can launch more than the default 20 instances. In the ticket, ask for a limit of 50 \( \text{t2.micro} \) instances and 20 \( \text{c4.large} \) instances in the region you are using.

**Note:** To deploy PCF to AWS GovCloud (US), log in to the AWS GovCloud (US) Console instead of the standard AWS Management Console and select the \( \text{us-gov-west-1} \) region.

**Note:** To deploy PCF to AWS China, set up an AWS China account and contact the Platform Architect assigned for your Pivotal account.

You can check the limits on your account by visiting the EC2 Dashboard on the AWS Management Console and clicking Limits on the left navigation.

Step 2: Create S3 Buckets

1. Navigate to the S3 Dashboard.

**Note:** S3 bucket names must be globally unique. When naming buckets, Pivotal recommends that you prefix the generic names below with an unique and helpfully identifiable string (i.e. ID-STRING-pcf-ops-manager-bucket, MY-IDENTIFIER-pcf-buildpacks-bucket, and so on). Then you should use the same prefix when naming other associated resources, such as IAM policies.

2. Perform the following steps to create five S3 buckets:
   - Click Create Bucket.
   - For Bucket name, enter ID-STRING-pcf-ops-manager-bucket.
   - For Region, select your region.
   - Click Next three times.
   - Click Create bucket.
   - Repeat the above steps to create four more S3 buckets: ID-STRING-pcf-buildpacks-bucket, ID-STRING-pcf-packages-bucket, ID-STRING-pcf-resources-bucket, and ID-STRING-pcf-droplets-bucket.

Step 3: Create an IAM User for PCF

Perform the following steps to create an Amazon Identity and Access Management (IAM) user with the minimal permissions necessary to run and install PCF:

1. Click IAM to access the IAM Dashboard.

2. Click Users and then click Add user.
3. Enter a user name, such as `pcf-user`.

4. For AWS access type, select **Programmatic access**.

   ![Note: If you prefer to create your keys locally and import them into AWS, see the Amazon documentation.](image)

5. Click **Next: Permissions**.

6. Click **Next: Review** and review your choices.

   ![Note: On the Review page you may see a warning that the user has no permissions. You can disregard this message. You do not need to set user permissions.](image)

7. Click **Create user**.

8. Click **Download .csv** to download the user security credentials.

   ![Warning: The credentials.csv contains the IDs for your user security access key and secret access key. Keep the credentials.csv file for your currently active key pairs in a secure directory. You cannot recover a lost key pair.](image)

9. Click **Close**.

10. On the **Users** page, click the user name to access the user details page.

    ![Note: On the Users page you may see a warning that the user has no permissions. You can disregard this message. You do not need to set user permissions.](image)
11. Click Add inline policy. You can review your existing inline policies by clicking the down arrow.

12. On the Create policy page, define a policy:
   a. Copy the policy document included in the Pivotal Cloud Foundry for AWS Policy Document topic. You must edit the policy document so the names of the S3 buckets match the ones you created in Step 2: Create S3 Buckets.
   b. Paste the policy document into the JSON tab on the Create policy page.

13. Click Review policy.

14. In the Name field, enter `pcf-iam-policy`.

15. Click Create policy. The Summary page displays a list of available policies and actions.

Step 4: Create a VPC

1. Navigate to the VPC Dashboard.

2. Click Start VPC Wizard.

3. Select VPC with Public and Private Subnets and click Select.
4. Specify the following details for your VPC:

- IPv4 CIDR block: Enter `10.0.0.0/16`.
- IPv6 CIDR block: Select `No IPv6 CIDR Block`.
- VPC name: `[pcf-vpc]`.
- Public subnet's IPv4 CIDR: Enter `10.0.0.0/24`.
- Set the Availability Zone fields for both subnets to `[REGION-a]`. For example, `us-west-2a`.
- Public subnet name: Enter `[pcf-public-subnet-az0]`.
- Private subnet's IPv4 CIDR: Enter `10.0.16.0/28`.
- Private subnet name: Enter `[pcf-management-subnet-az0]`.
- Click **Use a NAT instance instead** and do the following:
  - Under Specify the details of your NAT instance, set the Instance type to `t2.medium`.
  - Create a key pair titled `[pcf-ops-manager-key]`. For more information about creating the key pair, see Amazon EC2 Key Pairs in the AWS documentation.
  - Select your newly-created `[pcf-ops-manager-key]` for the Key Pair name.
  - Enable DNS hostnames: Click Yes.
  - Hardware tenancy: Select Default.
  - Click **Create VPC**.

5. After the VPC is successfully created, click **Subnets** in the left navigation.

6. Click **Create Subnet**.

7. Add the following subnets to the `[pcf-vpc]` VPC:

<table>
<thead>
<tr>
<th>Name</th>
<th>AZ</th>
<th>IPv4 CIDR block</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pcf-public-subnet-az1</code></td>
<td><code>[REGION-b]</code></td>
<td><code>10.0.1.0/24</code></td>
</tr>
<tr>
<td><code>pcf-public-subnet-az2</code></td>
<td><code>[REGION-c]</code></td>
<td><code>10.0.2.0/24</code></td>
</tr>
<tr>
<td><code>pcf-management-subnet-az1</code></td>
<td><code>[REGION-b]</code></td>
<td><code>10.0.16.0/28</code></td>
</tr>
<tr>
<td><code>pcf-management-subnet-az2</code></td>
<td><code>[REGION-c]</code></td>
<td><code>10.0.16.12/28</code></td>
</tr>
<tr>
<td><code>pcf-ert-subnet-az0</code></td>
<td><code>[REGION-a]</code></td>
<td><code>10.0.4.0/24</code></td>
</tr>
<tr>
<td><code>pcf-ert-subnet-az1</code></td>
<td><code>[REGION-b]</code></td>
<td><code>10.0.5.0/24</code></td>
</tr>
<tr>
<td><code>pcf-ert-subnet-az2</code></td>
<td><code>[REGION-c]</code></td>
<td><code>10.0.6.0/24</code></td>
</tr>
<tr>
<td><code>pcf-services-subnet-az0</code></td>
<td><code>[REGION-a]</code></td>
<td><code>10.0.8.0/24</code></td>
</tr>
<tr>
<td><code>pcf-services-subnet-az1</code></td>
<td><code>[REGION-b]</code></td>
<td><code>10.0.9.0/24</code></td>
</tr>
<tr>
<td><code>pcf-services-subnet-az2</code></td>
<td><code>[REGION-c]</code></td>
<td><code>10.0.10.0/24</code></td>
</tr>
<tr>
<td><code>pcf-rds-subnet-az0</code></td>
<td><code>[REGION-a]</code></td>
<td><code>10.0.12.0/24</code></td>
</tr>
<tr>
<td><code>pcf-rds-subnet-az1</code></td>
<td><code>[REGION-b]</code></td>
<td><code>10.0.13.0/24</code></td>
</tr>
<tr>
<td><code>pcf-rds-subnet-az2</code></td>
<td><code>[REGION-c]</code></td>
<td><code>10.0.14.0/24</code></td>
</tr>
</tbody>
</table>

**Note:** You created the first two subnets in the previous step: `[pcf-public-subnet-az0]` and `[pcf-management-subnet-az0]`. 
Step 5: Configure a Security Group for Ops Manager

1. Return to the EC2 Dashboard.


3. For Security group name, enter pcf-ops-manager-security-group.

4. For Description, enter a description to identify this security group.

5. For VPC, select the VPC where you want to deploy Ops Manager.

6. Click the Inbound tab and add rules according to the table below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>My IP</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>My IP</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>My IP</td>
</tr>
<tr>
<td>BOSH Agent</td>
<td>TCP</td>
<td>6868</td>
<td>10.0.0.0/16</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

7. Click Create.

Step 6: Configure a Security Group for PCF VMs

1. From the Security Groups page, click Create Security Group to create another security group.

2. For Security group name, enter pcf-vms-security-group.

3. For Description, enter a description to identify this security group.

4. For VPC, select the VPC where you want to deploy the PCF VMs.

5. Click the Inbound tab and add rules for all traffic from your public and private subnets to your private subnet, as the table and image show. This rule configuration does the following:
   - Enables BOSH to deploy PAS and other services.
   - Enables application VMs to communicate through the router.
   - Allows the load balancer to send traffic to Pivotal Application Service (PAS).

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>0-65535</td>
<td>Custom IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

6. Click Create.

Step 7: Configure a Security Group for the Web ELB

1. From the Security Groups page, click Create Security Group to create another security group.
2. For **Security group name**, enter `pcf-web-elb-security-group`.

3. For **Description**, enter a description to identify this security group.

4. For **VPC**, select the VPC where you want to deploy this Elastic Load Balancer (ELB).

5. Click the **Inbound** tab and add rules to allow traffic to ports 80, 443, and 4443 from 0.0.0.0/0, as the table and image show.

   ![Note: Allow traffic to port 4443 only if you are in an AWS cloud region that does not support AWS ALBs. For example, the GovCloud region. For more information about AWS regions and availability zones, see AWS Global Infrastructure.](image)

   ![Note: For finer control over what can reach PAS, change 0.0.0.0 to be more restrictive. This security group governs external access to PAS from apps such as the cf CLI and app URLs.](image)

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP rule</td>
<td>TCP</td>
<td>4443</td>
<td>Anywhere</td>
</tr>
<tr>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>Anywhere</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>Anywhere</td>
</tr>
</tbody>
</table>

6. Click **Create**.

---

**Step 8: Configure a Security Group for the SSH ELB**

1. From the **Security Groups** page, click **Create Security Group** to create another security group.

2. For **Security group name**, enter `pcf-ssh-elb-security-group`.

3. For **Description**, enter a description to identify this security group.

4. For **VPC**, select the VPC where you want to deploy this ELB.

5. Click the **Inbound** tab and add the following rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP rule</td>
<td>TCP</td>
<td>2222</td>
<td>Anywhere</td>
</tr>
</tbody>
</table>

6. Click **Create**.
Step 9: Configure a Security Group for the TCP ELB

1. From the Security Groups page, click Create Security Group to create another security group.

2. For Security group name, enter `pcf-tcp-elb-security-group`.

3. For Description, enter a description to identify this security group.

4. For VPC, select the VPC where you want to deploy this ELB.

5. Click the Inbound tab and add the following rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP rule</td>
<td>TCP</td>
<td>1024 - 1123</td>
<td>Anywhere</td>
</tr>
</tbody>
</table>

6. Click Create.

Step 10: Configure a Security Group for the Outbound NAT

1. From the Security Groups page, click Create Security Group to create another security group.

2. For Security group name, enter `pcf-nat-security-group`.

3. For Description, enter a description to identify this security group.

4. For VPC, select the VPC where you want to deploy the Outbound NAT.

5. Click the Inbound tab and add a rule to allow all traffic from your VPCs, as the table and image show.

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>Custom IP</td>
</tr>
</tbody>
</table>

6. Click Create.

Step 11: Configure a Security Group for MySQL

Note: If you plan to use an internal database, skip this step. If you are using RDS, you must configure a security group that enables the Ops Manager VM and BOSH Director VM to access the database.

1. From the Security Groups page, click Create Security Group to create another security group.

2. For Security group name, enter `pcf-mysql-security-group`.

3. For Description, enter a description to identify this security group.

4. For VPC, select the VPC where you want to deploy MySQL.

5. Click the Inbound tab. Add a rule of type `MySQL` and specify the subnet of your VPC in Source, as the table and image show.

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>TCP</td>
<td>3306</td>
<td>Custom IP</td>
</tr>
</tbody>
</table>
6. Click the **Outbound** tab. Add a rule of type **All traffic** and specify the subnet of your VPC in **Destination**, as the table and image show.

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>Custom IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

7. Click **Create**.

![Security group rules]

**Step 12: Launch an Ops Manager AMI**

To launch an Amazon Machine Image (AMI) for Ops Manager, do the following:

1. Navigate to the **Pivotal Cloud Foundry Operations Manager** section of **Pivotal Network**.
2. Select the version of PCF you want to install from the **Releases** dropdown.
3. In the **Release Download Files**, click the file named **Pivotal Cloud Foundry Ops Manager for AWS** to download a PDF.
4. Open the PDF and identify the AMI ID for your region.
5. Return to the EC2 Dashboard.
6. Click **AMIs** from the **Images** menu.
7. Select **Public images** from the drop-down filter that says **Owned by me**.
8. Paste the AMI ID for your region into the search bar and press enter.

---

**Note:** There is a different AMI for each region. If you cannot locate the AMI for your region, verify that you have set your AWS Management Console to your desired region. If you still cannot locate the AMI, log in to the **Pivotal Network** and file a support ticket.

9. (Optional) If you want to encrypt the VM that runs Ops Manager with AWS Key Management Service (KMS), perform the following additional steps:
   a. Right click the row that lists your AMI and click **Copy AMI**.
   b. Select your **Destination region**.
   c. Enable **Encryption**. For more information about AMI encryption, see **Encryption and AMI Copy** from the Copying an AMI topic in the AWS documentation.
   d. Select your **Master Key**. To create a new custom key, see **Creating Keys** in the AWS documentation.
   e. Click **Copy AMI**. You can use the new AMI you copied for the following steps.
10. Select the row that lists your Ops Manager AMI and click **Launch**.
11. Choose **m3.large** for your instance type and click **Next: Configure Instance Details**.

© Copyright Pivotal Software Inc, 2013-2019
12. Configure the following for your instance:
- **Network**: Select the VPC that you created.
- **Subnet**: Select `pcf-public-subnet-az0` to allow traffic from public IP addresses, or `pcf-management-subnet-az0` to allow traffic only from private IP addresses.
- **Auto-assign for Public IP**: Select **Enable** to allow traffic from public IP addresses, or **Disable** to allow traffic only from private IP addresses.
- **IAM role**: Select the IAM role associated with your `pcf-user` profile. If you have not created one, click **Create new IAM role** and follow the [Guidelines for Creating User Roles on AWS](#).
- For all other fields, accept the default values.

13. Click **Next: Add Storage** and adjust the **Size (GiB)** value. The default persistent disk value is 50 GB. Pivotal recommends increasing this value to a minimum of 100 GB.

14. Click **Next: Tag Instance**.

15. On the **Add Tags** page, add a tag with the key **Name** and value `pcf-ops-manager`.

16. Click **Next: Configure Security Group**.

17. Select the `pcf-ops-manager-security-group` that you created in **Step 5: Configure a Security Group for Ops Manager**.
18. Click Review and Launch and confirm the instance launch details.

19. Click Launch.

20. Select the pcf-ops-manager-key key pair, confirm that you have access to the private key file, and click Launch Instances. You use this key pair to access the Ops Manager VM.

21. Click View Instances to access the Instances page on the EC2 Dashboard.

Step 13: Create Web Load Balancer

1. On the EC2 Dashboard, click Load Balancers.

2. Click Create Load Balancer.

3. Under Application Load Balancer, click Create.

4. For Step 1: Configure Load Balancer, do the following:
   a. Under Basic Configuration, do the following:
      - For Name, enter pcf-web-elb.
      - For Scheme, select internet-facing to allow traffic from public IP addresses, or internal to allow traffic only from private IP addresses.
      - For IP address type, select the type of IP addresses you want to allow.
   b. Under Listeners, click Add listener. For Load Balancer Protocol, select HTTPS.
   c. Under Availability Zones, select all availability zones.
   d. Click Next: Configure Security Settings.

5. For Step 2: Configure Security Settings, do the following:
   a. Under Select default certificate, do one of the following:
      - If you already have a certificate from AWS Certificate Manager (ACM), select Choose a certificate from ACM.
      - If you do not have a certificate from ACM, select Upload a certificate to ACM. For more information, see Importing Certificates into AWS Certificate Manager in the AWS documentation.

      Note: For a production or production-like environment, use a certificate from a Certificate Authority (CA). This can be an internal certificate or a purchased certificate. For a sandbox environment, you can use a self-signed certificate.
   b. For Certificate Name, select the desired certificate.
   c. For Security Policy, select the policy you created in Step 3: Create an IAM User for PCF.
   d. Click Next: Configure Security Groups.

6. For Step 3: Configure Security Groups, do the following:
   a. Under Assign a security group, select Select an existing security group.
   b. From the list of security groups, select the pcf-web-elb-security-group security group that you configured in Step 7: Configure a Security Group for the Web ELB.
c. Click Next: Configure Routing.

7. For Step 4: Configure Routing, do the following:
   a. Under Target Group, enter the following values:
      - Name: Enter `pcf-web-elb-target-group`.
      - Protocol: Select HTTP.
   b. Under Health checks, set Path to `/health`.
   c. Under Advanced health check settings, enter the following values:
      - Port: Set to 8080.
      - Interval: Set to 5 seconds.
      - Timeout: Set to 3 seconds.
      - Unhealthy threshold: Set to 3.
      - Health threshold: Set to 6.
   d. Click Next: Register Targets.

8. For Step 5: Register Targets, accept the default values and click Next: Review.

9. For Step 6: Review, review the load balancer details and then click Create. A message appears to confirm AWS successfully created the load balancer.

Step 14: Create SSH Load Balancer

1. From the Load Balancers page, click Create Load Balancer.

2. Select Classic Load Balancer.

3. Configure the load balancer with the following information:
   - Load Balancer name: Enter `pcf-ssh-elb`.
   - Create LB Inside: Select the `pcf-vpc` VPC that you created in Step 4: Create a VPC.
   - If you want to allow traffic from public IP addresses, ensure that the Create an internal load balancer checkbox is not selected. If you want to allow traffic only from private IP addresses, select this checkbox.
4. Under Listener Configuration, add the following rules:

<table>
<thead>
<tr>
<th>Load Balancer Protocol</th>
<th>Load Balancer Port</th>
<th>Instance Protocol</th>
<th>Instance Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>2222</td>
<td>TCP</td>
<td>2222</td>
</tr>
</tbody>
</table>

5. Under Select Subnets, select either the public or private subnets you configured in Step 4: Create a VPC, and click Next: Assign Security Groups.


7. On the Configure Security Settings page, ignore the Improve your load balancer’s security error message and click Next: Configure Health Check.

8. On the Configure Health Check page, enter the following values:
   - Ping Protocol: Select `TCP`.
   - Ping Port: Set to `2222`.

© Copyright Pivotal Software Inc, 2013-2019
- Interval: Set to 5 seconds.
- Response Timeout: Set to 3 seconds.
- Unhealthy threshold: Set to 3.
- Health threshold: Set to 6.

9. Click Next: Add EC2 Instances.

10. Accept the defaults on the Add EC2 Instances page and click Next: Add Tags.

11. Accept the defaults on the Add Tags page and click Review and Create.

12. Review and confirm the load balancer details, and click Create.

**Step 15: Create TCP Load Balancer**

1. From the Load Balancers page, click Create Load Balancer.

2. Select Classic Load Balancer.

3. Configure the load balancer with the following information:
   - Load Balancer name: Enter pcf-tcp-elb.
   - Create LB Inside: Select the pcf-vpc VPC that you created in Step 4: Create a VPC.
   - If you want to allow traffic from public IP addresses, ensure that the Create an internal load balancer checkbox is not selected. If you want to allow traffic only from private IP addresses, select this checkbox.
4. Under **Listener Configuration**, add the following rules:

<table>
<thead>
<tr>
<th>Load Balancer Protocol</th>
<th>Load Balancer Port</th>
<th>Instance Protocol</th>
<th>Instance Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>1024</td>
<td>TCP</td>
<td>1024</td>
</tr>
<tr>
<td>TCP</td>
<td>1025</td>
<td>TCP</td>
<td>1025</td>
</tr>
<tr>
<td>TCP</td>
<td>1026</td>
<td>TCP</td>
<td>1026</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>TCP</td>
<td>1123</td>
<td>TCP</td>
<td>1123</td>
</tr>
</tbody>
</table>

The entry above indicates that you must add listening rules for each port between 1026 and 1123.

5. Under **Select Subnets**, select either the public or private subnets you configured in **Step 4: Create a VPC**, and click Next: **Assign Security Groups**.

6. On the **Assign Security Groups** page, select the security group `pcf-tcp-elb-security-group` you configured in **Step 9: Configure a Security Group for the TCP ELB**, and click Next: **Configure Security Settings**.
7. On the Configure Security Settings page, ignore the Improve you load balancer’s security error message and click Next: Configure Health Check.

8. On the Configure Health Check page, enter the following values:
   - Ping Protocol: Select TCP.
   - Ping Port: Set to 80.
   - Interval: Set to 5 seconds.
   - Response Timeout: Set to 3 seconds.
   - Unhealthy threshold: Set to 3.
   - Health threshold: Set to 6.

9. Click Next: Add EC2 Instances.

10. Accept the defaults on the Add EC2 Instances page and click Next: Add Tags.

11. Accept the defaults on the Add Tags page and click Review and Create.

12. Review and confirm the load balancer details, and click Create.

Step 16: Configure DNS Records

1. Perform the following steps for all three of the load balancers you created in previous steps, named pcf-web-elb, pcf-ssh-elb, and pcf-tcp-elb:
   a. From the Load Balancers page, select the load balancer.
   b. On the Description tab, locate the Basic Configuration section and record the DNS name of the load balancer.

2. Click Instances on the left navigation to view your EC2 instances

3. Select the PcfOpsManInstance instance created by Cloudformation.

4. On the Description tab, record the value for IPv4 Public IP.
5. Navigate to your DNS provider and create the following CNAME and A records:

- **CNAME**: *.apps.YOUR-SYSTEM-DOMAIN.com and *.system.YOUR-SYSTEM-DOMAIN.com points to the DNS name of the pcf-web-elb load balancer.
- **CNAME**: ssh.YOUR-SYSTEM-DOMAIN.com points to the DNS name of the pcf-ssh-elb load balancer.
- **CNAME**: tcp.YOUR-SYSTEM-DOMAIN.com points to the DNS name of the pcf-tcp-elb load balancer.
- **A**: pcf.YOUR-SYSTEM-DOMAIN.com points to the public IP address of the pcf-ops-manager EC2 instance.

**Step 17: Secure the NAT Instance**

1. On the EC2 Dashboard, click **Instances**.
2. Select the NAT instance, which has an instance type of **t2.medium**.
3. From the **Actions** menu, select **Networking>Change Security Groups**.
4. Change the NAT security group from the default group to the **pcf-nat-security-group** NAT security group that you created in **Step 10: Configure a Security Group for the Outbound NAT**.
5. Click **Assign Security Groups**.

**Step 18: Create RDS Subnet Group**

1. Navigate to the RDS Dashboard.
2. Perform the following steps to create a RDS Subnet Group for the two RDS subnets:
   a. Click **Subnet Groups>Create DB Subnet Group**.
   b. Enter the following values:
      - **Name**: Enter **pcf-rds-subnet-group**.
      - **Description**: Enter a description to identify this subnet group.
      - **VPC ID**: Select **pcf-vpc**.
      - **Availability Zone and Subnet ID**: Choose the AZ and subnet for **pcf-rds-subnet-az0** and click **Add**.
      - Repeat the steps above to add **pcf-rds-subnet-az1** and **pcf-rds-subnet-az2** to the group.
   d. Click **Create**.

The following screenshot shows a completed subnet group.
Step 19: Create a MySQL Database Using AWS RDS

**Note:** You must have an empty MySQL database when you install or reinstall PCF on AWS.

1. Navigate to the RDS Dashboard.
2. Click **Instances>Launch DB Instance** to launch the wizard.
3. Select **MySQL**.
4. Select the **MySQL** radio button under **Production** to create a database for production environments.
5. Click **Next Step**.
6. Specify the following database details:
   - **DB Instance Class**: Select **db.m3.large** - 2 vCPU, 7.5 GiB RAM.
   - **Multi-AZ Deployment**: Select **Yes**.
   - **Storage Type**: Select **Provisioned IOPS (SSD)**.
   - **Allocated Storage**: Enter **100 GB**.
   - **DB Instance Identifier**: Enter **pcf-ops-manager-director**.
   - Enter a secure **Master Username** and **Master Password**.

**Note:** Record the username and password. You need these credentials later when configuring the **Director Config** page in the BOSH Director tile.
7. Click Next Step.

8. On the Configure Advanced Settings page, enter the following values:
   - **VPC**: Select **pcf-vpc**.
   - **Subnet Group**: Select the **pcf-rds-subnet-group** you created in Step 18: Create RDS Subnet Group.
   - **Publicly Accessible**: Select **No**.
   - **VPC Security Groups**: Select the **pcf-rds-security-group** that you created in Step 11: Configure a Security Group for MySQL.
   - **Database Name**: Enter `bosh`.
   - Accept the default values for the remaining fields.
9. Click Launch DB Instance. Launching the instance may take several minutes.

10. When the instance has launched, proceed to Configuring BOSH Director on AWS to continue deploying PCF.

Required AWS Objects

This section describes the AWS objects you create in the procedures above in order to deploy PCF.

Use this section to determine the resource requirements of PCF on AWS, or to verify that you created the correct resources after completing the procedures above.
S3 Buckets for Ops Manager and PAS

You must create the following S3 buckets from the S3 Dashboard:

- `pcf-ops-manager-bucket`
- `pcf-builpacks-bucket`
- `pcf-packages-bucket`
- `pcf-resources-bucket`
- `pcf-droplets-bucket`

These buckets must be empty when you install or reinstall PCF.

See [Step 2: Create S3 Buckets](#).

IAM User for PCF

You must create an IAM user for PCF named `pcf-user` from the Identity and Access Management Dashboard, using the policy document included in the Pivotal Cloud Foundry for AWS Policy Document topic.

See [Step 3: Create an IAM User for PCF](#).

Key Pair

You must generate a key pair named `pcf-ops-manager-key`. For more information about setting up a key pair, see [Amazon EC2 Key Pairs](#) in the AWS documentation.

VPC (Public and Private Subnets)

You must create a VPC with public and private subnets from the VPC Dashboard.

The following table lists the subnets in CIDR block `10.0.0.0/16`:

<table>
<thead>
<tr>
<th>Name</th>
<th>AZ</th>
<th>IPv4 CIDR block</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pcf-public-subnet-az0</code></td>
<td>REGION-#a</td>
<td>10.0.0.0/24</td>
</tr>
<tr>
<td><code>pcf-public-subnet-az1</code></td>
<td>REGION-#b</td>
<td>10.0.1.0/24</td>
</tr>
<tr>
<td><code>pcf-public-subnet-az2</code></td>
<td>REGION-#c</td>
<td>10.0.2.0/24</td>
</tr>
<tr>
<td><code>pcf-management-subnet-az0</code></td>
<td>REGION-#a</td>
<td>10.0.16.0/28</td>
</tr>
<tr>
<td><code>pcf-management-subnet-az1</code></td>
<td>REGION-#b</td>
<td>10.0.16.16/28</td>
</tr>
<tr>
<td><code>pcf-management-subnet-az2</code></td>
<td>REGION-#c</td>
<td>10.0.16.32/28</td>
</tr>
<tr>
<td><code>pcf-ert-subnet-az0</code></td>
<td>REGION-#a</td>
<td>10.0.4.0/24</td>
</tr>
<tr>
<td><code>pcf-ert-subnet-az1</code></td>
<td>REGION-#b</td>
<td>10.0.5.0/24</td>
</tr>
<tr>
<td><code>pcf-ert-subnet-az2</code></td>
<td>REGION-#c</td>
<td>10.0.6.0/24</td>
</tr>
<tr>
<td><code>pcf-services-subnet-az0</code></td>
<td>REGION-#a</td>
<td>10.0.8.0/24</td>
</tr>
<tr>
<td><code>pcf-services-subnet-az1</code></td>
<td>REGION-#b</td>
<td>10.0.9.0/24</td>
</tr>
<tr>
<td><code>pcf-services-subnet-az2</code></td>
<td>REGION-#c</td>
<td>10.0.10.0/24</td>
</tr>
<tr>
<td><code>pcf-rds-subnet-az0</code></td>
<td>REGION-#a</td>
<td>10.0.12.0/24</td>
</tr>
<tr>
<td><code>pcf-rds-subnet-az1</code></td>
<td>REGION-#b</td>
<td>10.0.13.0/24</td>
</tr>
<tr>
<td><code>pcf-rds-subnet-az2</code></td>
<td>REGION-#c</td>
<td>10.0.14.0/24</td>
</tr>
</tbody>
</table>

See [Step 4: Create a VPC](#).
NAT Instance

You must create a NAT instance when creating a VPC. The NAT instance must have the following configuration:

- **Instance type**: t2.medium
- **Key pair name**: pcf-ops-manager-key
- **Enable DNS hostnames**: Yes
- **Hardware tenancy**: Default

See Step 4: Create a VPC.

You must also assign the NAT instance to the pcf-nat-security-group. See Step 17: Secure the NAT Instance.

Security Groups

The following sections describe the security groups you must create from the EC2 Dashboard.

Ops Manager

The Ops Manager Security Group must be named pcf-ops-manager-security-group and have the following inbound rules:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>My IP</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>My IP</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>My IP</td>
</tr>
<tr>
<td>BOSH Agent</td>
<td>TCP</td>
<td>6868</td>
<td>10.0.0.0/16</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

See Step 5: Configure a Security Group for Ops Manager.

PCF VMs

The PCF VMs Security Group must be named pcf-vms-security-group and have the following inbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>0 - 65535</td>
<td>Custom IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

See Step 6: Configure a Security Group for PCF VMs.

Web ELB

The Web ELB Security Group must be named pcf-web-elb-security-group and have the following inbound rules:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP rule</td>
<td>TCP</td>
<td>4443</td>
<td>Anywhere</td>
</tr>
<tr>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>Anywhere</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>Anywhere</td>
</tr>
</tbody>
</table>

See Step 7: Configure a Security Group for the Web ELB.

SSH ELB

The SSH ELB Security Group must be named pcf-ssh-elb-security-group and have the following inbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The SSH ELB Security Group must have the following outbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>Anywhere</td>
<td>0.0.0.0/0</td>
</tr>
</tbody>
</table>

See [Step 8: Configure a Security Group for the SSH ELB](#).

TCP ELB

The TCP ELB Security Group must be named `pcf-tcp-elb-security-group` and have the following inbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP rule</td>
<td>TCP</td>
<td>1024 - 1123</td>
<td>Anywhere</td>
</tr>
</tbody>
</table>

The TCP ELB Security Group must have the following outbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>Anywhere</td>
<td>0.0.0.0/0</td>
</tr>
</tbody>
</table>

See [Step 9: Configure a Security Group for the TCP ELB](#).

Outbound NAT

The Outbound NAT Security Group must be named `pcf-nat-security-group` and have the following inbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>Custom IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

See [Step 10: Configure a Security Group for the Outbound NAT](#).

MySQL

The MySQL Security Group must be named `pcf-mysql-security-group` and have the following inbound rules:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>TCP</td>
<td>3306</td>
<td>Custom IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

The MySQL Security Group must have the following outbound rules:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>Custom IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10.0.0.0/16</td>
</tr>
</tbody>
</table>

See [Step 11: Configure a Security Group for MySQL](#).

Ops Manager AMI

You must locate the public Ops Manager AMI using the AMI ID provided by the PDF downloaded when clicking [Pivotal Cloud Foundry Ops Manager for AWS](#) on Pivotal Network.

See [Step 12: Launch a Pivotal Ops Manager AMI](#).

ELBs

The following sections describe the ELBs you must create from the EC2 Dashboard.
Web ELB

You must create a web ELB with the following configuration:

- Name: `pcf-web-elb`
- LB Inside: `pcf-vpc`
- Selected Subnet: `pcf-public-subnet-az0`, `pcf-public-subnet-az1`, `pcf-public-subnet-az2`
- Security Group: `pcf-elb-security-group`
- Health Check: TCP Port 8080, Path: `/health`


SSH ELB

- Name: `pcf-ssh-elb`
- LB Inside: `pcf-vpc`
- Selected Subnet: `pcf-public-subnet-az0`, `pcf-public-subnet-az1`, `pcf-public-subnet-az2`
- Security Group: `pcf-ssh-security-group`
- Health Check: TCP Port 2222

See Step 14: Create SSH Load Balancer.

TCP ELB

- Name: `pcf-tcp-elb`
- LB Inside: `pcf-vpc`
- Selected Subnet: `pcf-public-subnet-az0`, `pcf-public-subnet-az1`, `pcf-public-subnet-az2`
- Security Group: `pcf-tcp-security-group`
- Health Check: TCP Port 80

See Step 15: Create TCP Load Balancer.

DNS Configuration

You must navigate to your DNS provider and create CNAME and A records for all three of your load balancers.

See Step 16: Configure DNS Records.

RDS Subnet Group

You must create a subnet group for RDS named `pcf-rds-subnet-group` from the RDS Dashboard.

See Step 18: Create RDS Subnet Group.

MySQL Database

You must create a MySQL database from the RDS Dashboard.

See Step 19: Create a MySQL Database using AWS RDS.
Configuring BOSH Director on AWS

This topic describes how to configure Ops Manager to deploy Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS).

Before beginning this procedure, ensure that you have successfully completed all of the steps in the Installing PCF on AWS Manually topic.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain you created in the Configure DNS Records step of Installing PCF on AWS Manually.

2. When Ops Manager starts for the first time, you must choose one of the following:

- **Internal Authentication**: If you use Internal Authentication, Ops Manager maintains your user database.
- **SAML Identity Provider**: If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
- **LDAP Server**: If you use a LDAP Server, an external identity server maintains your user database.

### Internal Authentication

1. When redirected to the Internal Authentication page, do the following:

   - Enter a Username, Password, and Password confirmation to create an Admin user.
   - Enter a Decryption passphrase and the Decryption passphrase confirmation. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an HTTP proxy or HTTPS proxy, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the End User License Agreement, and select the checkbox to accept the terms.
   - Click Setup Authentication.
2. Log in to Ops Manager with the Admin username and password you created in the previous step.

SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.
3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **6a.** Ops Manager SAML service provider metadata: [https://OPS-MAN-FQDN:443/uaa/saml/metadata](https://OPS-MAN-FQDN:443/uaa/saml/metadata)
   - **6b.** BOSH Director SAML service provider metadata: [https://BOSH-IP-ADDRESS:8443/saml/metadata](https://BOSH-IP-ADDRESS:8443/saml/metadata)

   **Note:** To retrieve your **BOSH-IP-ADDRESS**, navigate to the **Status** tab in the **BOSH Director** tile. Record the **BOSH Director** IP address.

7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Audience URI (SP Entity ID)**: [https://OP-MAN-FQDN:443/uaa](https://OP-MAN-FQDN:443/uaa)
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on URL**: [https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP](https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP)
   - **Audience URI (SP Entity ID)**: [https://BOSH-IP:8443](https://BOSH-IP:8443)
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

### LDAP Server

- **Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - `ldap://`: This specifies that the LDAP server uses an unencrypted connection.
  - `ldaps://`: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or...
that you import a trusted certificate to the JVM truststore.

- For **LDAP Username** and **LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: cn=administrator,ou=Users,dc=example,dc=com

  **Note:** Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

**warning:** Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: ou=Users,dc=example,dc=com

- For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter \[ cn=Smith \] returns all objects with a common name equal to Smith.

  In the LDAP search filter string that you use to configure your runtime, use \[ {0} \] instead of the username. For example, use \[ cn={0} \] to return all LDAP objects with the same common name as the username.

  **Note:** Other attributes commonly searched for and returned are \[ mail \], \[ uid \], and, in the case of Active Directory, \[ sAMAccountName \].

- For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

  For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: ou=Groups,dc=example,dc=com

- For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is \[ member={0} \].

- For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example \[ mail \].

- For **LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.

- From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
  - Automatically follow any referrals.
  - Ignore referrals and return partial result.
  - Throw exception for each referral and abort.

- For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

- Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

- If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

- Read the **End User License Agreement**, and select the checkbox to accept the terms.

- Select **Provision an admin client in the BOSH UAA**. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

- Click **Setup Authentication**.

- Return to the **BOSH Director** tile, and continue with the configuration steps below.
Step 2: AWS Config Page

1. Click the BOSH Director tile.

2. Select AWS Config.

   ![AWS Management Console Config](image)

   - Use AWS Keys
     - Access Key ID
     - AWS Secret Key
   - Use AWS Instance Profile
     - AWS IAM Instance Profile
     - Security Group ID
     - Key Pair Name
     - SSH Private Key
   - Region
   - Encrypt Linux EBS Volumes
     - Custom Encryption Key

3. Select Use AWS Keys or Use AWS Instance Profile.
- **Access Key ID and AWS Secret Key**: To retrieve your AWS key information, use the AWS Identity and Access Management (IAM) credentials that you generated in the Step 3: Create an IAM User for PCF section of the Installing PCF on AWS Manually topic.
- **AWS IAM Instance Profile**: Enter the name of the IAM profile you created in the Step 3: Create an IAM User for PCF section of the Installing PCF on AWS Manually topic.

4. Complete the rest of the AWS Management Console Config page:
   - **Security Group ID**: Enter the Group ID of the `pcf-vms-security-group` you created for your PCF VMs in the Step 6: Configure a Security Group for PCF VMs section of the Installing PCF on AWS Manually topic. Locate the Group ID in the Security Groups tab of your EC2 Dashboard.
   - **Key Pair Name**: Enter `pcf-ops-manager-key`.
   - **SSH Private Key**: Open the AWS key pair `pcf-ops-manager-keys.pem` file you generated in the Step 3: Create an IAM User for PCF section of the Installing PCF on AWS Manually topic. Copy the contents of the `.pem` file and paste it into the SSH Private Key field.
   - **Region**: Select the region where you deployed Ops Manager.
   - **Encrypt Linux EBS Volumes**: Select this checkbox to enable full encryption on persistent disks of all BOSH-deployed VMs except the Ops Manager VM and Director VM. See the Configuring Amazon EBS Encryption for PCF on AWS topic for details about using EBS encryption.

   **Note**: Enabling EBS encryption only encrypts Linux VMs. The Windows VMs deployed with Pivotal Application Service (PAS) for Windows or PAS for Windows 2012R2 are not encrypted.

5. Click **Save**.

---

## Step 3: Director Config Page

1. In Ops Manager, select **Director Config**.

![Director Config](image)

   - **NTP Servers (comma delimited)**
   - **JMX Provider IP Address**
   - **Bosh HM Forwarder IP Address**
   - **Enable VM Resurrecter Plugin**
   - **Enable Post Deploy Scripts**
   - **Recreate All VMs**
   - **Recreate All Persistent Disks**
   - **Enable bosh deploy retries**

   This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.

   Checking this box will recreate all Persistent Disks for the Director and all other Tiles

   This will attempt to re-deploy a failed deployment up to 5 times.

2. Enter at least two of the following NTP servers in the **NTP Servers (comma delimited)** field, separated by a comma:
   - `0.amazon.pool.ntp.org,1.amazon.pool.ntp.org,2.amazon.pool.ntp.org,3.amazon.pool.ntp.org`

   **Note**: The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify the value of this field.

3. Leave the **JMX Provider IP Address** field blank.

4. Leave the **Bosh HM Forwarder IP Address** field blank.
5. Select the **Enable VM Resurrection Plugin** checkbox to enable the BOSH Resurrection functionality and increase your runtime availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   **Note:** If you intend to install, you must enable post-deploy scripts.

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. (Optional) Disable **Allow Legacy Agents** if all of your tiles have stemcells 3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.

13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


   ![HM Pager Duty Plugin](image)

   - **Service Key:** Enter your API service key from PagerDuty.
   - **HTTP Proxy:** Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

---

**Note:** Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank.

---

**Note:** If you intend to install, you must enable post-deploy scripts.
Host: Enter your email hostname.
Port: Enter your email port number.
Domain: Enter your domain.
From: Enter the address for the sender.
Recipients: Enter comma-separated addresses of intended recipients.
Username: Enter the username for your email server.
Password: Enter the password for your email server.
Enable TLS: Select this checkbox to enable Transport Layer Security to the email host.

16. For CredHub Encryption Provider, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

Before configuring an HSM encryption provider in the Director Config pane, you must follow the procedures and collect information described in Preparing CredHub HSMs for Configuration.

Note: After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
- **Internal**: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.
- **Luna HSM**: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

  1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
  2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in [Preparing CredHub HSMs for Configuration](#).
  3. **Provider Partition Password**
  4. **Provider Client Certificate**
  5. **Provider Client Certificate Private Key**
  6. **HSM Host Address**
  7. **HSM Port Address**: If you do not know your port address, enter 1792.
  8. **Partition Serial Number**
  9. **HSM Certificate**: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. Select a **Blobstore Location** to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable and less secure, Pivotal recommends that you configure an external blobstore.
Internal: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.

Enable TLS: Select this checkbox to enable TLS to the blobstore.

Note: If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

S3 Compatible Blobstore: Select this option to use an external S3-compatible endpoint. When you have created an S3 bucket, complete the following steps:

1. S3 Endpoint: Navigate to the Regions and Endpoints topic in the AWS documentation.
   a. Locate the endpoint for your region in the Amazon Simple Storage Service (S3) table and construct a URL using your region’s endpoint. For example, if you are using the us-west-2 region, the URL you create would be https://s3-us-west-2.amazonaws.com. Enter this URL into the S3 Endpoint field.
   b. On a command line, run ssh ubuntu@OPS-MANAGER-FQDN to SSH into the Ops Manager VM. Replace OPS-MANAGER-FQDN with the fully qualified domain name of Ops Manager.

Note: After you deploy Ops Manager, you cannot change the blobstore location.
c. Copy the custom public CA certificate you used to sign the S3 endpoint into `/etc/ssl/certs` on the Ops Manager VM.

d. On the Ops Manager VM, run `sudo update-ca-certificates -f -v` to import the custom CA certificate into the Ops Manager VM truststore.

**Note:** You must also add this custom CA certificate into the **Trusted Certificates** field in the **Security** page. See Security Page for instructions.

2. **Bucket Name**: Enter the name of the S3 bucket.
3. **Access Key** and **Secret Key**: Enter the keys you generated when creating your S3 bucket.
4. **Select V2 Signature** or **V4 Signature**. If you select **V4 Signature**, enter your **Region**.

**Note:** AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see Authenticating Requests in the AWS documentation.

- **GCS Blobstore**: Select this option to use an external GCS endpoint. To create a GCS bucket, you must have a GCS account. Follow the procedures in Creating Storage Buckets in the GCS documentation to create a GCS bucket. When you have created a GCS bucket, complete the following steps:

  1. **Bucket Name**: Enter the name of your GCS bucket.
  2. **Storage Class**: Select the storage class for your GCS bucket. See Storage Classes in the GCP documentation for more information.
  3. **Service Account Key**: Follow the steps in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually to download a JSON file with a private key. Enter the contents of the JSON file into the field.

18. For **Database Location**, select **External MySQL Database** and complete the following steps:

- From the AWS Console, navigate to the RDS Dashboard.
- Select **Instances**, then click the arrow to the left of your instance and select the second icon to display the **Details** information.
- Refer to the following table to retrieve the values for the **Director Config** page:

<table>
<thead>
<tr>
<th>RDS Instance Field</th>
<th>BOSH Director Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>Host</td>
</tr>
<tr>
<td>Port</td>
<td>Port, which is 3306</td>
</tr>
<tr>
<td>DB Name</td>
<td>Database, which is bosh</td>
</tr>
<tr>
<td>Username</td>
<td>Username</td>
</tr>
</tbody>
</table>

- For **Password**, enter the password that you defined for your MySQL database when you created it in the Step 19: Create a MySQL Database using AWS RDS section of the Installing PCF on AWS Manually topic. In addition, if you selected the **Enable TLS for Director Database** checkbox, you can fill out the following optional fields:
  - **Enable TLS**: Selecting this checkbox enables TLS communication between the BOSH Director and the database.
  - **TLS CA**: Enter the Certificate Authority for the TLS Certificate.
- **TLS Certificate**: Enter the client certificate for mutual TLS connections to the database.
- **TLS Private Key**: Enter the client private key for mutual TLS connections to the database.
- **Advanced DB Connection Options**: If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.

19. (Optional) Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to 5.

20. (Optional) **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. (Optional) To add a custom URL for your BOSH Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director in the Pivotal Support Knowledge Base.

22. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

23. (Optional) Enter your comma-separated custom **Identification Tags**. For example, `iaas:foundation1, hello:world`. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

24. Click **Save**.

### Step 4: Create Availability Zones Page

- **Note**: Pivotal recommends at least three availability zones (AZs) for a highly available installation of PAS. The procedures in Installing PCF on AWS Manually use 3 AZs.

1. Select **Create Availability Zones**.
2. To add the three AZs you specified in the Step 4: Create a VPC section of the Installing PCF on AWS Manually topic, do the following:
   a. Click Add.
   b. For Amazon Availability Zone, enter the name of the AZ.
   c. Repeat until you have entered all three AZs, in the format REGION-#a, REGION-#b, and REGION-#c. For example, us-west-2a, us-west-2b, and us-west-2c.

3. Click Save.

Step 5: Create Networks Page

1. Select Create Networks.

2. (Optional) Select Enable ICMP checks to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. Perform the following steps to add the network configuration that you created for your VPC in the Create a VPC section of Installing PCF on AWS Manually. Record your VPC CIDR if you set a CIDR other than the recommendation!
   a. Click Add Network.
b. For Name, enter infrastructure.

c. Create a subnet for each availability zone by clicking Add Subnet. Refer to the table below for the information required to create all three subnets:

<table>
<thead>
<tr>
<th>First Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-infrastructure-subnet-az0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.16.0/28</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.16.0-10.0.16.4</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2*</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.16.1</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>REGION-#a. Example: us-west-2a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Second Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-infrastructure-subnet-az1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.16.16/28</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.16.16-10.0.16.20</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2*</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.16.17</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>REGION-#b. Example: us-west-2b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Third Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-infrastructure-subnet-az2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.16.32/28</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.16.32-10.0.16.36</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2*</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.16.33</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>REGION-#c. Example: us-west-2c</td>
</tr>
</tbody>
</table>

* If you set a VPC CIDR other than recommended, enter the second IP in your VPC CIDR. For example, for a 10.0.0.0/24 VPC CIDR, enter 10.0.0.2 in each subnet.

d. Click Add Network.

e. For Name, enter the name of your runtime. For example, pas or pks.

f. Create a subnet for each availability zone by clicking Add Subnet. See the table below for the information required to create all three subnets:

<table>
<thead>
<tr>
<th>First Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-pas-subnet-az0 or pcf-pks-subnet-az0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.4.0/24</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.4.0-10.0.4.4</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2*</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.4.1</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>REGION-#a. Example: us-west-2a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Second Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-pas-subnet-az1 or pcf-pks-subnet-az1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.5.0/24</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.5.0-10.0.5.4</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2*</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.5.1</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>REGION-#b. Example: us-west-2b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Third Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-pas-subnet-az2 or pcf-pks-subnet-az2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.6.0/24</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.6.0-10.0.6.4</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2*</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.6.1</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>REGION-#c. Example: us-west-2c</td>
</tr>
</tbody>
</table>

* If you set a VPC CIDR other than recommended, enter the second IP in your VPC CIDR. For example, for a 10.0.0.0/24 VPC CIDR, enter 10.0.0.2 in each subnet.

g. Click Add Network.
h. For Name, enter services.

i. Create a subnet for each availability zone by clicking Add Subnet. Refer to the table below for the information required to create all three subnets:

<table>
<thead>
<tr>
<th>First Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-services-subnet-az0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDR</td>
<td>10.0.8.0/24</td>
<td></td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>10.0.8.0-10.0.8.3</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>10.0.0.2 *</td>
<td></td>
</tr>
<tr>
<td>Gateway</td>
<td>10.0.8.1</td>
<td></td>
</tr>
<tr>
<td>Availability Zones</td>
<td>REGION-#a</td>
<td>Example: us-west-2a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Second Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-services-subnet-az1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDR</td>
<td>10.0.9.0/24</td>
<td></td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>10.0.9.0-10.0.9.3</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>10.0.0.2 *</td>
<td></td>
</tr>
<tr>
<td>Gateway</td>
<td>10.0.9.1</td>
<td></td>
</tr>
<tr>
<td>Availability Zones</td>
<td>REGION-#b</td>
<td>Example: us-west-2b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Third Subnet</th>
<th>VPC Subnet ID</th>
<th>pcf-services-subnet-az2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDR</td>
<td>10.0.10.0/24</td>
<td></td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>10.0.10.0-10.0.10.3</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>10.0.0.2 *</td>
<td></td>
</tr>
<tr>
<td>Gateway</td>
<td>10.0.10.1</td>
<td></td>
</tr>
<tr>
<td>Availability Zones</td>
<td>REGION-#c</td>
<td>Example: us-west-2c</td>
</tr>
</tbody>
</table>

* If you set a VPC CIDR other than recommended, enter the second IP in your VPC CIDR. For example, for a 10.0.0.0/24 VPC CIDR, enter 10.0.0.2 in each subnet.

4. Click Save.

**Note:** If you are deploying with a workload load balancer, you must tag each AWS subnet with your Kubernetes cluster unique identifier before you create the load balancer. For more information about tagging subnets with a cluster unique identifier, see AWS Prerequisites.

**Note:** After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see Expanding Your Network with Additional Subnets.

---

**Step 6: Assign AZs and Networks**

1. Select Assign AZs and Networks

   ![Assign AZs and Networks](image-url)

The BOSH Director is a single instance.

Choose the availability zone in which to place that instance. It is highly recommended that you back up this VM on a regular basis to preserve settings.

Singleton Availability Zone

Network

Infrastructure

Save

---

© Copyright Pivotal Software Inc, 2013-2019 139 2.5
2. Use the dropdown to select a **Singleton Availability Zone**. The BOSH Director is deployed into this AZ.

3. Use the dropdown to select **infrastructure** under **Network**. The BOSH Director is deployed into this network.

4. Click **Save**.

### Step 7: Security Page

1. Select **Security**.

![Security Page](image)

2. In **Trusted Certificates**, enter your custom certificate authority (CA) certificates to insert into your organization’s certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEFGH12345678ABCDEFGH12345678ABCDEFGH12345678ABCD

-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
BCDEFGH12345678ABCDEFGH12345678ABCDEFGH12345678ABCDEF

-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
CDEFGH12345678ABCDEFGH12345678ABCDEFGH12345678ABCDEF

-----END CERTIFICATE-----
```

3. **(Optional)** Select the **Include OpsManager Root CA in Trusted Certs** checkbox to include the Ops Manager root CA in the Trusted Certificates field. The BOSH Director includes this CA in the trust store of every VM that it deploys.

**Note:** If you want to use Docker registries to run PAS app instances in Docker containers, enter the certificate for your private Docker registry in this field. See [Using Docker Registries](#) for more information about running app instances in PAS using Docker registries.

4. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater
security.

5. Click Save. To view your saved Director password, click the Credentials tab.

Step 8: BOSH DNS Config Page

1. Select BOSH DNS Config.

2. (Optional) In Excluded Recursors, enter a list of prohibited recursor addresses.

3. (Optional) In Recursor Timeout, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   ![Note: This time limit must include one of the Go parse duration time units. For example, entering 5s sets the timeout limit to five seconds. For more information about supported time units, see `func ParseDuration` in the Go Programming Language documentation.]

4. (Optional) In Handlers, enter a list of custom domain handlers in JSON format.

5. Click Save.

Step 9: Syslog Page
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.
7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for syslog in the **Custom rsyslog Configuration** field. This field requires the rsyslogscript syntax.

10. Click **Save Syslog Settings**.

**Step 10: Resource Config Page**

1. Select **Resource Config**.

2. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the dropdown to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

   **Note:** Pivotal recommends provisioning a BOSH Director VM with at least 8 GB memory.

   **Note:** If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the new recommended allocation.

   **Note:** If you install PAS for Windows, provision your **Master Compilation Job** with at least 100 GB of disk space.

3. (Optional) Enter your AWS target group name in the **Load Balancers** column for each job. Prepend the name with **alb:**. For example, enter **alb:target-group-name**. To create an Application Load Balancer (ALB) and target group, follow the procedures in [Getting Started with Application Load Balancers](#) in the AWS documentation. Then navigate to **Target Groups** in the EC2 Dashboard menu to find your target group **Name**.

   **Note:** To configure an ALB, you must have the following AWS IAM permissions.

   ```
   "elasticloadbalancing:DescribeLoadBalancers",
   "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
   "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
   "elasticloadbalancing:DescribeTargetGroups",
   "elasticloadbalancing:RegisterTargetGroups"
   ```

4. Click **Save**.

**Step 11: Complete the BOSH Director Installation**

1. Return to the **Installation Dashboard**.

2. Click **Review Pending Changes**, then **Apply Changes**.
3. BOSH Director begins to install. The Changes Applied window displays when the installation process successfully completes.

Next Steps

After completing the procedures in this topic, you must configure a runtime for PCF. You can install PCF on AWS with the Pivotal Application Service (PAS) or runtime.

To configure a runtime, do one of the following:

- Configure PAS. See Configuring PAS.
- Configure . See Installing on AWS.
Installing PCF on AWS using Terraform

This topic explains how to install PCF on AWS using Terraform.

Overview

You can install PCF on AWS with either the Pivotal Application Service (PAS) or Pivotal Container Service (PKS) runtime.

To install PCF on AWS using Terraform, do one of the following:

- Install with PAS. See Install with PAS Using Terraform.
- Install with PKS. See Install with PKS Using Terraform.

Install with PAS Using Terraform

To install with PAS using Terraform, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on AWS Using Terraform.
2. Configure BOSH Director. See Configuring BOSH Director on AWS Using Terraform.
3. Configure PAS. See Configuring PAS.

Install with PKS Using Terraform

To install with PKS using Terraform, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on AWS Using Terraform.
2. Configure BOSH Director. See Configuring BOSH Director on AWS Using Terraform.
3. Configure PKS. See Installing PKS on AWS.
Deploying Ops Manager on AWS Using Terraform

Page last updated:

This guide describes the preparation steps required to deploy Ops Manager on Amazon Web Services (AWS) using Terraform templates.

The Terraform template for Ops Manager on AWS describes a set of AWS resources and properties. For more information about how Terraform creates resources in AWS, see the AWS Provider topic on the Terraform site.

If you are deploying Pivotal Application Service (PAS), you may also find it helpful to review different deployment options in the Reference Architecture for Pivotal Cloud Foundry on AWS.

Prerequisites

Before you deploy Ops Manager on AWS, review the following:

- If you intend to install PAS, see PCF on AWS Requirements.
- If you intend to install, see AWS Prerequisites and Resource Requirements.

In addition to reviewing the prerequisites for your runtime, ensure you have the following:

- The Terraform CLI
- In your AWS account, ensure you have an IAM user with the following permissions:
  - AmazonEC2FullAccess
  - AmazonRDSFullAccess
  - AmazonRoute53FullAccess
  - AmazonS3FullAccess
  - AmazonVPCFullAccess
  - IAMFullAccess
  - AWSKeyManagementServicePowerUser

Step 1: Download Templates and Edit Variables File

Before you can run Terraform commands to provision infrastructure resources, you must download the AWS Terraform templates and create a Terraform template variables file as described below:

2. Download the AWS Terraform templates ZIP file.
3. Extract the contents of the ZIP file.
4. Move the extracted folder to the workspace directory on your local machine.
5. On the command line, navigate to the directory. For example:

   
   $ cd ~/workspace/pivotal-cf-terraforming-aws

6. Navigate to the terraforming-pas or terraforming-pks directory that contains the Terraform files for your runtime.
7. In the runtime directory, create a text file named terraform.tfvars.
8. Open the terraform.tfvars file and add the following:
9. Edit the values in the file according to the table below.

<table>
<thead>
<tr>
<th>Value to replace</th>
<th>Guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOUR-ENVIRONMENT-NAME</td>
<td>Enter a name to use to identify resources in AWS. Terraform prepends the names of the resources it creates with this environment name. Examples: ['pcf', 'pas', 'pks'].</td>
</tr>
<tr>
<td>YOUR-ACCESS-KEY</td>
<td>Enter your AWS Access Key ID of the AWS account in which you want Terraform to create resources.</td>
</tr>
<tr>
<td>YOUR-SECRET-KEY</td>
<td>Enter your AWS Secret Access Key of the AWS account in which you want Terraform to create resources.</td>
</tr>
<tr>
<td>YOUR-AWS-REGION</td>
<td>Enter the name of the AWS region in which you want Terraform to create resources. Example: us-central-1.</td>
</tr>
<tr>
<td>YOUR-AZ-1, YOUR-AZ-2, YOUR-AZ-3</td>
<td>Enter three availability zones from your region. Example: ['us-central-1a', 'us-central-1b', 'us-central-1c'].</td>
</tr>
<tr>
<td>YOUR-OPS-MAN-IMAGE-AMI</td>
<td>Enter the source code for the Ops Manager Amazon Machine Image (AMI) you want to boot. You can find this code in the PDF included with the Ops Manager release on Pivotal Network. If you want to encrypt your Ops Manager VM, create an encrypted AMI copy from the AWS EC2 dashboard and enter the source code for the copied Ops Manager image instead. For more information about copying an AMI, see step 9 of Launch an Ops Manager AMI in the manual AWS configuration topic.</td>
</tr>
<tr>
<td>YOUR-DNS-SUFFIX</td>
<td>Enter a domain name to use as part of the system domain for your deployment. Terraform creates DNS records in AWS using YOUR-ENVIRONMENT-NAME and YOUR-DNS-SUFFIX. For example, if you enter example.com for your DNS suffix and have pcf as your environment name, Terraform will create DNS records at pcf.example.com.</td>
</tr>
<tr>
<td>YOUR-CERTIFICATE</td>
<td>Enter a certificate to use for HTTP load balancing. For production environments, use a certificate from a Certificate Authority (CA). For test environments, you can use a self-signed certificate. Your certificate must specify your system domain as the common name. Your system domain is YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX. It also must include the following subdomains: *.sys.YOUR-SYSTEM-DOMAIN, *.login.sys.YOUR-SYSTEM-DOMAIN, *.ui.sys.YOUR-SYSTEM-DOMAIN, *.apps.YOUR-SYSTEM-DOMAIN.</td>
</tr>
<tr>
<td>YOUR-PRIVATE-KEY</td>
<td>Enter a private key for the certificate you entered.</td>
</tr>
</tbody>
</table>

Step 2: Add Optional Variables

Complete this step if you want to do any of the following in PAS:

- Use an AWS Relational Database Service (RDS) for your deployment. For more information, see Getting started with Amazon RDS in Amazon RDS resources.
- Deploy the Isolation Segment tile.
In your `terraform.tfvars` file, specify the appropriate variables from the sections below.

Note: You can see the configurable options by opening the `variables.tf` file and looking for variables with default values.

### Isolation Segments
If you plan to deploy the Isolation Segment tile, add the following variables to your `terraform.tfvars` file, replacing `YOUR-CERTIFICATE` and `YOUR-PRIVATE-KEY` with a certificate and private key. This causes terraform to create an additional HTTP load balancer across three availability zones to use for the Isolation Segment tile.

```bash
create_isoseg_resources = 1
iso_seg_ssl_cert = "-----BEGIN CERTIFICATE-----
YOUR-CERTIFICATE
-----END CERTIFICATE-----
ISO_SEG_SSL_CERT"
iso_seg_ssl_cert_private_key = "-----BEGIN EXAMPLE RSA PRIVATE KEY-----
YOUR-PRIVATE-KEY
-----END EXAMPLE RSA PRIVATE KEY-----
ISO_SEG_SSL_KEY"
```

### RDS
1. If you want to use an RDS for Ops Manager and PAS, add the following to your `terraform.tfvars` file:

```bash
rds_instance_count = 1
```

2. If you want to specify a username for RDS authentication, add the following variable to your `terraform.tfvars` file.

```bash
rds_db_username = username
```

### Step 3: Create AWS Resources with Terraform
Follow these steps to use the Terraform CLI to create resources on AWS:

1. From the directory that contains the Terraform files, run `terraform init` to initialize the directory based on the information you specified in the `terraform.tfvars` file.

   ```bash
   terraform init
   ```

2. Run `terraform plan -out=plan` to create the execution plan for Terraform.

   ```bash
   terraform plan -out=plan
   ```

3. Run `terraform apply plan` to execute the plan from the previous step. It may take several minutes for Terraform to create all the resources in AWS.

   ```bash
   terraform apply plan
   ```

### Step 4: Create DNS Record
1. In a browser, navigate to the DNS provider for the DNS suffix you entered in your `terraform.tfvars` file.

2. Create a new NS record for your system domain. Your system domain is: `YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX`.

3. In this record, enter the name servers included in `env_dns_zone_name_servers` from your Terraform output.

© Copyright Pivotal Software Inc, 2013-2019
What to Do Next

Proceed to the next step in the deployment, Configuring BOSH Director on AWS Using Terraform.
Configuring BOSH Director on AWS Using Terraform

This topic describes how to configure the BOSH Director tile in Ops Manager on Amazon Web Services (AWS) after [Deploying Ops Manager on AWS Using Terraform](#).

**Prerequisites**

To complete the procedures in this topic, you must have access to the output from when you ran `terraform apply` to create resources for this deployment. You can view this output at any time by running `terraform output`. You use the values in your Terraform output to configure the BOSH Director tile.

**Step 1: Access Ops Manager**

1. In a web browser, navigate to the fully qualified domain name (FQDN) of Ops Manager. Use the `ops_manager_dns` value from running `terraform output`. You use the values in your Terraform output to configure the BOSH Director tile.

2. When Ops Manager starts for the first time, you must choose one of the following:
   - **Internal Authentication**: If you use Internal Authentication, Ops Manager maintains your user database.
   - **SAML Identity Provider**: If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
   - **LDAP Server**: If you use a LDAP Server, an external identity server maintains your user database.

   ![Diagram of the Ops Manager authentication types]

**Internal Authentication**

1. When redirected to the Internal Authentication page, do the following:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
If you are using an HTTP proxy or HTTPS proxy, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.

Read the End User License Agreement, and select the checkbox to accept the terms.

Click Setup Authentication.

2. Log in to Ops Manager with the Admin username and password you created in the previous step.

SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.
3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.

- **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
- **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:

   - **6a.** Ops Manager SAML service provider metadata: https://OPS-MAN-FQDN:443/uaa/saml/metadata
   - **6b.** BOSH Director SAML service provider metadata: https://BOSH-IP-ADDRESS:8443/saml/metadata

   **Note:** The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager login page.

7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.

   - **Single sign on URL**: https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN
   - **Audience URI (SP Entity ID)**: https://OP-MAN-FQDN:443/uaa
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.

   - **Single sign on URL**: https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP
   - **Audience URI (SP Entity ID)**: https://BOSH-IP:8443
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

9. Return to the BOSH Director tile, and continue with the configuration steps below.

### LDAP Server
- **For Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - **ldap://**: This specifies that the LDAP server uses an unencrypted connection.
  - **ldaps://**: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or
that you import a trusted certificate to the JVM truststore.

- For **LDAP Username** and **LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: 
  
  \texttt{cn=administrator,ou=Users,dc=example,dc=com}

  \textbf{Note:} Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

\begin{warning}
\textbf{warning:} Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.
\end{warning}

- For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: \texttt{ou=Users,dc=example,dc=com}

- For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter \texttt{cn=Smith} returns all objects with a common name equal to \texttt{Smith}.

  In the LDAP search filter string that you use to configure your runtime, use \texttt{[0]} instead of the username. For example, use \texttt{cn=[0]} to return all LDAP objects with the same common name as the username.

  In addition to \texttt{cn}, other attributes commonly searched for and returned are \texttt{mail}, \texttt{uid}, and, in the case of Active Directory, \texttt{xSAMAccountName}.

  \textbf{Note:} For instructions for testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry in the Pivotal Support Knowledge Base.

- For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

  For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: \texttt{ou=Groups,dc=example,dc=com}

  \begin{itemize}
  \item For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is \texttt{member=[0]}.
  \item For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example \texttt{mail}.
  \item For **LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.
  \item From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
    \begin{itemize}
    \item Automatically follow any referrals.
    \item Ignore referrals and return partial result.
    \item Throw exception for each referral and abort.
    \end{itemize}
  \end{itemize}

- For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

- Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

- If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

- Read the **End User License Agreement**, and select the checkbox to accept the terms.

- Select **Provision an admin client in the BOSH UAA**. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

- Click **Setup Authentication**.

- Return to the **BOSH Director** tile, and continue with the configuration steps below.
Step 2: AWS Config Page

1. Click the **BOSH Director** tile.

2. Select **AWS Config** to open the **AWS Management Console Config** page.

3. Select **Use AWS Keys** or **Use AWS Instance Profile**.

© Copyright Pivotal Software Inc, 2013-2019
If you choose to use AWS keys, complete the following fields:

- **Access Key ID**: Enter the value of `ops_manager_iam_user_access_key` from the Terraform output.
- **AWS Secret Key**: Enter the value of `ops_manager_iam_user_secret_key` from the Terraform output.

If you choose to use an AWS instance profile, enter the name of your AWS Identity and Access Management (IAM) profile.

4. Complete the remainder of the **AWS Management Console Config** page with the following information.

- **Security Group ID**: Enter the value of `vms_security_group_id` from the Terraform output.
- **Key Pair Name**: Enter the value of `ops_manager_ssh_public_key_name` from the Terraform output.
- **SSH Private Key**: Run `terraform output` to view the value of `ops_manager_ssh_private_key`, and enter it into this field.
- **Region**: Select the region where you deployed Ops Manager.
- **Encrypt Linux EBS Volumes**: Select this checkbox to enable full encryption on persistent disks of all BOSH-deployed VMs, except for the Ops Manager VM and BOSH Director VM. See the [Configuring Amazon EBS Encryption](#) topic for details about using Elastic Block Store (EBS) encryption.

> Note: Enabling EBS encryption only encrypts Linux VMs. The Windows VMs deployed with Pivotal Application Service for Windows (PASW) are not encrypted.

- **Custom Encryption Key (Optional)** Once you enable EBS encryption, you may want to specify a custom Key Management Service (KMS) encryption key. If you don’t enter a value, your custom encryption key will default to the account key. For more information, see [Configuring Amazon EBS Encryption](#).

5. Click **Save**.

### Step 3: Director Config Page

1. In Ops Manager, select **Director Config**.

![Director Config](image)

- **NTP Servers (comma delimited)**
- **JMX Provider IP Address**
- **Bosh HM Forwarder IP Address**
- **Enable VM Resurrection Plugin**
- **Enable Post Deploy Scripts**
- **Recreate All VMs**
  - This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.
- **Recreate All Persistent Disks**
  - Checking this box will recreate all Persistent Disks for the Director and all other Tiles
- **Enable bosh deploy retries**
  - This will attempt to re-deploy a failed deployment up to 5 times.

2. Enter at least two of the following NTP servers in the **NTP Servers (comma delimited)** field, separated by a comma:

   `0.amazon.pool.ntp.org,1.amazon.pool.ntp.org,2.amazon.pool.ntp.org,3.amazon.pool.ntp.org`

> Note: The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify...
3. Leave the **JMX Provider IP Address** field blank.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

   ![Note: Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the Bosh HM Forwarder IP Address field blank.]

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the BOSH Resurrector functionality and increase your runtime availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   ![Note: If you intend to install, you must enable post-deploy scripts.]

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. (Optional) Disable **Allow Legacy Agents** if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.

13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


   ![HM Pager Duty Plugin]

   - **Service Key**: Enter your API service key from PagerDuty.
   - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.
- **Host**: Enter your email hostname.
- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in [Preparing CredHub HSMs for Configuration](#).

**Note:** After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
- **Internal**: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.
- **Luna HSM**: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in [Preparing CredHub HSMs for Configuration](#).
3. **Provider Partition Password**
4. **Provider Client Certificate**
5. **Provider Client Certificate Private Key**
6. **HSM Host Address**
7. **HSM Port Address**: If you do not know your port address, enter **1792**.
8. **Partition Serial Number**
9. **HSM Certificate**: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. Select a **Blobstore Location** to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable and less secure, Pivotal recommends that you configure an external blobstore.
Internal: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.

Enable TLS: Select this checkbox to enable TLS to the blobstore.

Note: If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

S3 Compatible Blobstore: Select this option to use an external S3-compatible endpoint. When you have created an S3 bucket, complete the following steps:

1. S3 Endpoint: Navigate to the Regions and Endpoints topic in the AWS documentation.

   a. Locate the endpoint for your region in the Amazon Simple Storage Service (S3) table and construct a URL using your region’s endpoint. For example, if you are using the us-west-2 region, the URL you create would be https://s3-us-west-2.amazonaws.com. Enter this URL into the S3 Endpoint field.

   b. On a command line, run ssh ubuntu@OPS-MANAGER-FQDN to SSH into the Ops Manager VM. Replace OPS-MANAGER-FQDN with the fully qualified domain name of Ops Manager.
c. Copy the custom public CA certificate you used to sign the S3 endpoint into /etc/ssl/certs on the Ops Manager VM.
d. On the Ops Manager VM, run `sudo update-ca-certificates -f -v` to import the custom CA certificate into the Ops Manager VM truststore.

```
Note: You must also add this custom CA certificate into the Trusted Certificates field in the Security page. See Security Page for instructions.
```

2. **Bucket Name**: Enter the name of the S3 bucket.
3. **Access Key** and **Secret Key**: Enter the keys you generated when creating your S3 bucket.
4. **Select V2 Signature or V4 Signature**: If you select V4 Signature, enter your Region.

```
Note: AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see Authenticating Requests in the AWS documentation.
```

- **GCS Blobstore**: Select this option to use an external GCS endpoint. To create a GCS bucket, you must have a GCS account. Follow the procedures in Creating Storage Buckets in the GCS documentation to create a GCS bucket. When you have created a GCS bucket, complete the following steps:
  1. **Bucket Name**: Enter the name of your GCS bucket.
  2. **Storage Class**: Select the storage class for your GCS bucket. See Storage Classes in the GCP documentation for more information.
  3. **Service Account Key**: Follow the steps in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually to download a JSON file with a private key. Enter the contents of the JSON file into the field.

```
Bucket Name: Enter the name of the S3 bucket.
Access Key and Secret Key: Enter the keys you generated when creating your S3 bucket.
Select V2 Signature or V4 Signature: If you select V4 Signature, enter your Region.
```

18. For **Database Location**, if you choose to configure an external MySQL database with Amazon Relational Database Service (RDS) or another service, select **External MySQL Database** and complete the fields below. Otherwise, select **Internal**. For more information about creating a RDS MySQL instance, see Creating a MySQL DB Instance in the AWS documentation.

```
- **Host**: Enter the value of your host.
- **Port**: Enter your port number. For example, `3306`.
- **Username**: Enter your username.
- **Password**: Enter your password.
- **Database**: Enter your database name.
In addition, if you selected the **Enable TLS for Director Database** checkbox, you can fill out the following optional fields:
  - **Enable TLS**: Selecting this checkbox enables TLS communication between the BOSH Director and the database.
  - **TLS CA**: Enter the Certificate Authority for the TLS Certificate.
  - **TLS Certificate**: Enter the client certificate for mutual TLS connections to the database.
  - **TLS Private Key**: Enter the client private key for mutual TLS connections to the database.
  - **Advanced DB Connection Options**: If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.
```

19. **(Optional)** Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to `5`.

20. **(Optional)** **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. **(Optional)** To add a custom URL for your BOSH Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director in the Pivotal Support Knowledge Base.

```
Director Workers
```

```
Max Threads
```

```
Director Hostname
```

22. **(Optional)** To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.
23. (Optional) Enter your comma-separated custom **Identification Tags**. For example, `iaas:foundation1, hello:world`. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

24. Click **Save**.

**Note:** For more information about AWS S3 Signatures, see the AWS [Authenticating Requests](#) documentation.

---

**Step 4: Create Availability Zones Page**

1. Select **Create Availability Zones**.

![Create Availability Zones](#)

2. To create three Availability Zones for your apps to use, do the following:
   a. Click **Add** three times.
   b. For **Amazon Availability Zone**, enter the values corresponding to the key `infrastructue_subnet_availability_zones` from the Terraform output.
   c. Click **Save**.

---

**Step 5: Create Networks Page**
1. Select Create Networks.

2. (Optional) Select Enable ICMP checks to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. To add the network configuration you created for your VPC, do the following:
   a. Click Add Network.
   b. For Name, enter infrastructure.
   c. Create a subnet for each availability zone by clicking Add Subnet. Refer to the table below for the information required to create all three subnets:

<table>
<thead>
<tr>
<th>First Subnet</th>
<th>VPC Subnet ID</th>
<th>The first value of infrastructure_subnet_ids from the Terraform output.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDR</td>
<td>10.0.16.0/28</td>
</tr>
<tr>
<td></td>
<td>Reserved IP Ranges</td>
<td>10.0.16.0-10.0.16.4</td>
</tr>
<tr>
<td></td>
<td>DNS</td>
<td>10.0.0.2 *</td>
</tr>
<tr>
<td></td>
<td>Gateway</td>
<td>10.0.16.1</td>
</tr>
<tr>
<td></td>
<td>Availability Zones</td>
<td>The first value of infrastructure_subnet_availability_zones from the Terraform output.</td>
</tr>
<tr>
<td></td>
<td>VPC Subnet ID</td>
<td>The second value of infrastructure_subnet_ids from the Terraform output.</td>
</tr>
</tbody>
</table>
Second Subnet
CIDR | 10.0.16.16/28
Reserved IP Ranges | 10.0.16.16-10.0.16.20
DNS | 10.0.0.2 *
Gateway | 10.0.16.17
Availability Zones | The second value of infrastructure_subnet_availability_zones from the Terraform output.

Third Subnet
CIDR | 10.0.16.32/28
Reserved IP Ranges | 10.0.16.32-10.0.16.36
DNS | 10.0.0.2 *
Gateway | 10.0.16.33
Availability Zones | The third value of infrastructure_subnet_availability_zones from the Terraform output.

* If you set a VPC CIDR other than recommended, enter the second IP in your VPC CIDR. For example, for a 10.0.0.0/24 VPC CIDR, enter 10.0.0.2 in each subnet.

d. Click Add Network.
e. For Name, enter the name of your runtime. For example, pas or pks.
f. Create a subnet for each availability zone by clicking Add Subnet. Refer to the table below for the information required to create all three subnets:

First Subnet
CIDR | 10.0.4.0/24
Reserved IP Ranges | 10.0.4.0-10.0.4.4
DNS | 10.0.0.2 *
Gateway | 10.0.4.1
Availability Zones | The first value of RUNTIME_subnet_availability_zones from the Terraform output.

Second Subnet
CIDR | 10.0.5.0/24
Reserved IP Ranges | 10.0.5.0-10.0.5.4
DNS | 10.0.0.2 *
Gateway | 10.0.5.1
Availability Zones | The second value of RUNTIME_subnet_availability_zones from the Terraform output.

Third Subnet
CIDR | 10.0.6.0/24
Reserved IP Ranges | 10.0.6.0-10.0.6.4
DNS | 10.0.0.2 *
Gateway | 10.0.6.1
Availability Zones | The third value of RUNTIME_subnet_availability_zones from the Terraform output.

* If you set a VPC CIDR other than recommended, enter the second IP in your VPC CIDR. For example, for a 10.0.0.0/24 VPC CIDR, enter 10.0.0.2 in each subnet.
g. Click Add Network.
h. For Name, enter services.
i. Create a subnet for each availability zone by clicking Add Subnet. Refer to the table below for the information required to create all three subnets:

First Subnet
CIDR | 10.0.8.0/24
Reserved IP Ranges | 10.0.8.0-10.0.8.3
DNS | 10.0.0.2 *

© Copyright Pivotal Software Inc, 2013-2019
### Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.

2. Use the dropdown to select a **Singleton Availability Zone**. The BOSH Director installs in this availability zone (AZ).

3. Use the dropdown to select the **infrastructure** network for your BOSH Director.

4. Click **Save**.

### Step 7: Security Page

1. Select **Security**.

---

*If you set a VPC CIDR other than recommended, enter the second IP in your VPC CIDR. For example, for a 10.0.0.0/24 VPC CIDR, enter 10.0.0.2 in each subnet.*

**Note:** If you are deploying with a workload load balancer, you must tag each AWS subnet with your Kubernetes cluster unique identifier before you create the load balancer. For more information about tagging subnets with a cluster unique identifier, see [AWS Prerequisites](#).

**Note:** After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see [Expanding Your Network with Additional Subnets](#).
2. In **Trusted Certificates**, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEFGHIJKLMNOPQRSTUVWXYZ

-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
ABCDEFGHIJKLMNOPQRSTUVWXYZ

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
ABCDEFGHIJKLMNOPQRSTUVWXYZ

-----END CERTIFICATE-----
```

3. (Optional) Select the **Include OpsManager Root CA in Trusted Certs** checkbox to include the OpsManager root CA in the Trusted Certificates field. BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

5. Click **Save**. To view your saved Director password, click the **Credentials** tab.

**Step 8: BOSH DNS Config Page**
1. Select **BOSH DNS Config**.

2. (Optional) In **Excluded Recursors**, enter a list of prohibited recursive addresses.

3. (Optional) In **Recursor Timeout**, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   **Note**: This time limit must include one of the Go parse duration time units. For example, entering `5s` sets the timeout limit to five seconds.

   For more information about supported time units, see `func ParseDuration` in the Go Programming Language documentation.

4. (Optional) In **Handlers**, enter a list of custom domain handlers in JSON format.

5. Click **Save**.

**Step 9: Syslog Page**
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.
7. (Optional) Enter an integer in Queue Size. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to Forward Debug Logs to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the Custom rsyslog Configuration field. This field requires the rainerscript syntax.

10. Click Save Syslog Settings.

**Step 10: Resource Config Page**

1. Select Resource Config.

2. Adjust any values as necessary for your deployment. Under the Instances, Persistent Disk Type, and VM Type fields, choose Automatic from the dropdown to allocate the recommended resources for the job. If the Persistent Disk Type field reads None, the job does not require persistent disk space.

   **Note:** Pivotal recommends provisioning a BOSH Director VM with at least 8 GB memory.

   **Note:** If you set a field to Automatic and the recommended resource allocation changes in a future version, Ops Manager automatically uses the new recommended allocation.

   **Note:** If you install PASW, provision your Master Compilation Job with at least 100 GB of disk space.

3. (Optional) Enter your AWS target group name in the Load Balancers column for each job. Prepend the name with alb:. For example, enter alb:target-group-name.

   To create an Application Load Balancer (ALB) and target group, follow the procedures in Getting Started with Application Load Balancers in the AWS documentation. Then navigate to Target Groups in the EC2 Dashboard menu to find your target group Name.

   **Note:** To configure an ALB, you must have the following AWS IAM permissions.

   ```
   "elasticloadbalancing:DescribeLoadBalancers",
   "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
   "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
   "elasticloadbalancing:DescribeTargetGroups",
   "elasticloadbalancing:RegisterTargets"
   ```

4. Click Save.

**Step 11: (Optional) Add Custom VM Extensions**

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see Managing Custom VM Extensions.
Step 11: Complete the BOSH Director Installation

1. Click the Installation Dashboard link to return to the Installation Dashboard.

2. Click Review Pending Changes, then Apply Changes. If the following ICMP error message appears, click Ignore errors and start the install.

3. BOSH Director installs. This may take a few moments. When the installation process successfully completes, the Changes Applied window appears.

4. After you complete this procedure, continue to the topic for the runtime you are installing:
   - PAS: Configuring PAS
   - **v1.2**: Installing on AWS CF
   - **v1.3**: Installing on AWS CF
Deleting PCF from AWS

When you deploy Pivotal Cloud Foundry (PCF) to Amazon Web Services (AWS), you provision a set of resources. This topic describes how to delete the AWS resources associated with a PCF deployment. You can use the AWS console to remove an installation of all components, but retain the objects in your bucket for a future deployment.

1. Log into your AWS Console.

2. Navigate to your EC2 dashboard. Select Instances from the menu on the left side.

3. Terminate all your instances.
4. Select **Load Balancers**. Delete all load balancers.
5. From the AWS Console, select RDS.

6. Select Instances from the menu on the left side. Delete the RDS instances.

7. Select Create final Snapshot from the dropdown. Click Delete.
8. From the AWS Console, select VPC.

9. Select Your VPCs from the menu on the left. Delete the VPCs.

10. Check the box to acknowledge that you want to delete your default VPC. Click Yes, Delete.
Creating a Proxy ELB for Diego SSH

If you want to allow SSH connections to application containers, you may want to use an Elastic Load Balancer (ELB) as the SSH proxy.

Perform the steps below to create this ELB:

1. On the EC2 Dashboard, click **Load Balancers**.

2. Click **Create Load Balancer** and configure a classic load balancer with the following information:

   - Enter a load balancer name.
   - Create LB Inside: Select the *pcf-vpc* VPC where your PCF installation lives.
   - Ensure that the Create an internal load balancer checkbox is not selected.

3. Under **Load Balancer Protocol**, ensure that this ELB is listening on TCP port 2222 and forwarding to TCP port 2222.

4. Under **Select Subnets**, select the public subnet.

5. On the **Assign Security Groups** page, create a new Security Group. This Security Group should allow inbound traffic on TCP port 2222.
6. The Configure Security Settings page displays a security warning because your load balancer is not using a secure listener. You can ignore this warning.

7. Click Next: Configure Health Check.

8. Select TCP in Ping Protocol on the Configure Health Check page. Ensure that the Ping Port value is 2222 and set the Health Check Interval to 30 seconds.

9. Click Next: Add EC2 Instances.

10. Accept the defaults on the Add EC2 Instances page and click Next: Add Tags.

11. Accept the defaults on the Add Tags page and click Review and Create.

12. Review and confirm the load balancer details and click Create.

13. With your DNS service (for example, Amazon Route 53), create an ssh.system.YOUR-SYSTEM-DOMAIN DNS record that points to this ELB that you just created.
14. You can now use this ELB to the SSH Proxy of your Pivotal Application Service (PAS) installation.

15. In PAS, select Resource Config and enter the ELB that you just created in the Diego Brain row, under the Load Balancers column.
Upgrading Ops Manager Director on AWS

This topic describes how to upgrade Ops Manager Director for Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS). It corresponds to Step 2: Deploy Ops Manager in the Upgrading Pivotal Cloud Foundry topic.

In this procedure, you create a new Ops Manager VM instance that hosts the new version of Ops Manager. To upgrade, you export your existing Ops Manager installation into this new VM.

Retrieve the Ops Manager AMI ID

To retrieve the Amazon Machine Image (AMI) ID for Ops Manager, perform the following steps:

1. Navigate to the Pivotal Cloud Foundry Operations Manager section of Pivotal Network.
2. Select the version of PCF you want to install from the Releases dropdown.
3. In the Release Download Files, click the file named Pivotal Cloud Foundry Ops Manager for AWS to download a PDF.
4. Open the PDF and record the AMI ID for your region.

Launch the AMI

To launch the AMI ID for Ops Manager, perform the following steps:

1. Navigate to your EC2 Dashboard.
2. Click AMIs from the Images menu.
3. Select Public images from the drop-down filter that says Owned by me.
4. Paste the AMI ID for your region into the search bar and press enter.

5. (Optional) If you want to encrypt the VM that runs Ops Manager with AWS Key Management Service (KMS), perform the following additional steps:
   a. Right click the row that lists your AMI and click Copy AMI.
   b. Select your Destination region.
   c. Enable Encryption. For more information about AMI encryption, see Encryption and AMI Copy from the Copying an AMI topic in the AWS documentation.
   d. Select your Master Key. To create a new custom key, see Creating Keys in the AWS documentation.
   e. Click Copy AMI. You can use the new AMI you copied for the following steps.
6. Select the row that lists your Ops Manager AMI and click Launch.
7. Choose m5.large for your instance type and click Next: Configure Instance Details.
8. Configure the following for your instance:
   - **Network**: Select the VPC that you created.
   - **Subnet**: Select `pcf-public-subnet-az0`.
   - **Auto-assign for Public IP**: Select `Enable`.
   - **IAM role**: Select the IAM role associated with your `pcf-user` profile. If you have not created one, click `Create new IAM role` and follow the Guidelines for Creating User Roles on AWS.
   - For all other fields, accept the default values.

9. Click **Next: Add Storage** and adjust the **Size (GiB)** value. The default persistent disk value is 50 GB. Pivotal recommends increasing this value to a minimum of 100 GB.

10. Click **Next: Tag Instance**.

11. On the **Add Tags** page, add a tag with the key `Name` and value `pcf-ops-manager`.

12. Click **Next: Configure Security Group**.

13. Select the `pcf-ops-manager-security-group` that you created in Step 5: **Configure a Security Group for Ops Manager**.

14. Click **Review and Launch** and confirm the instance launch details.

15. Click **Launch**.

16. Select the `pcf-ops-manager-key` key pair, confirm that you have access to the private key file, and click **Launch Instances**. You use this key pair to
access the Ops Manager VM.

Edit the Ops Manager DNS A Record

1. Click View Instances to access the Instances page on the EC2 Dashboard.

2. Select the VM that you created in the previous section.

3. Locate the IPv4 Public IP value in the instance Description tab, and record this value for use in the next step.

4. In your DNS provider, edit the A record for `pcf.YOUR-SYSTEM-DOMAIN` to point to the IP address recorded in the previous step.

Continue Upgrading PCF

Continue to Step 3: Import Previous Ops Manager Installation in the Upgrading Pivotal Cloud Foundry topic.
Installing Pivotal Cloud Foundry on Azure

This topic describes how to install Pivotal Cloud Foundry (PCF) on Microsoft Azure.

It includes resource requirements, prerequisites, instructions for installing PCF on Azure, and additional resources.

Overview

You can install PCF on Azure, Azure Government Cloud, or Azure Germany. For more information about Azure regions, see Azure regions in the Microsoft Azure documentation.

You can install PCF on Azure with the Pivotal Application Service (PAS) runtime. There are resource requirements specific to both PAS and Azure. Ensure you meet the requirements for PAS and the requirements specific to Azure before installing PCF on Azure. For more information, see Requirements.

Requirements

This section lists the following resource requirements for installing PCF on Azure:

- PAS resource requirements. See PAS Resource Requirements.
- Azure-specific resource requirements. See Azure Resource Requirements.

PAS Resource Requirements

The following are general resource requirements for deploying and managing a PCF deployment with Ops Manager and PAS:

- PAS requires sufficient IP allocation. The following lists the minimum required IP allocations:
  - One static IP address for either HAProxy or one of your Gorouters
  - One static IP address for each job in the Ops Manager tile. See the Ops Manager Resource Config pane for each tile for a full list.
  - One static IP address for each job listed below:
    - Consul
    - NATS
    - File Storage
    - MySQL Proxy
    - MySQL Server
    - Backup Restore Node
    - HAProxy
    - Router
    - MySQL Monitor
    - Diego Brain
    - TCP Router
  - One IP for each VM instance created by the service.
  - An additional IP address for each compilation worker. Use the following formula to determine the total IPs required: 
    \[ \text{IPs needed} = \text{static IPs} + \text{VM instances} + \text{compilation workers} \]
  - Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.
  - Pivotal recommends using a network without DHCP for deploying PAS VMs.

  Note: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

Azure Resource Requirements

The following are the minimum resource requirements for deploying a PCF deployment with Ops Manager and PAS on Azure:

- An OS disk of 120 GB for the Ops Manager VM.
A minimum of the following VM instance limits in your Azure account. The number of VMs required depends on the number of tiles and availability zones you plan to deploy. The following VM guidelines apply to the PAS and Small Footprint PAS runtimes:

- **PAS**: A new Azure deployment requires the following VMs for PAS and Ops Manager:

<table>
<thead>
<tr>
<th>VM Type</th>
<th>VM Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1s</td>
<td>27</td>
</tr>
<tr>
<td>F2s</td>
<td>4</td>
</tr>
<tr>
<td>F4s</td>
<td>4</td>
</tr>
<tr>
<td>DS11 v2</td>
<td>1</td>
</tr>
<tr>
<td>DS12 v2</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:** If you are deploying a test or sandbox PCF that does not require high availability, then you can scale down the number of VM instances in your deployment. For more information, see [Scaling PAS](#).

- **Small Footprint PAS**: A new Azure deployment requires the following VMs for Small Footprint PAS and Ops Manager:

<table>
<thead>
<tr>
<th>VM Type</th>
<th>VM Count</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS11 v2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DS12 v2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>F2s</td>
<td>0</td>
<td>Add 1 to count if using HAProxy</td>
</tr>
<tr>
<td>F1s</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>DS2 v2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>F4s</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Some VM instance types are only supported in certain Azure regions. For more information, see [Products available by region](#) in the Azure documentation. To deploy PCF on an Azure region that does not support the above VM instance types, you must override the default VM sizes. For more information, see [Overriding defaults with custom disk types](#) in the Ops Manager API documentation. Changing the default VM sizes may increase the cost of your deployment.

### Prerequisites

Before installing PCF on Azure, you must do the following:

- **Install the Azure CLI v2.0.** For instructions on how to install the Azure CLI for your operating system, see [Preparing to Deploy Ops Manager on Azure Manually](#).
- **Configure sufficient IP allocation for PAS.** For more information about IP allocation requirements, see [PAS Resource Requirements](#).
- **Create an SSL certificate for your PCF domain.**

**Note:** To deploy PCF to a production environment, you must obtain a certificate from a certificate authority. Pivotal recommends using a self-signed certificate generated by Ops Manager for development and testing purposes only.

- **Assign administrative rights to a domain for PCF.** You must be able to add wildcard records to this domain.
- **Create a wildcard DNS record that points to your router or load balancer.** Alternatively, you can use a service such as xip.io. For example, 203.0.113.0.xip.io. Then, create at least one wildcard TLS certificate that matches the DNS record you configured.

**Note:** With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer. For example, if you create a DNS record *.example.com pointing to your router, every app deployed to the example.com domain resolves to the IP address of your router.

- **Create one or more NTP servers, if the NTP servers are not already provided by your Azure project.**
- **Install the most recent version of the Cloud Foundry Command Line Interface (cf CLI).** See [Cloud Foundry Command Line Interface](#) in GitHub.
- **(Optional) Configure external storage.** Pivotal recommends using external storage if possible. For more information about how file storage location affects platform performance and stability during upgrades, see [Configure File Storage](#).
Configure external databases. Pivotal recommends using external databases in production deployments for BOSH Director and PAS. An external database must be configured to use the UTC timezone.

(Optional) Configure external user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.

Install PCF on Azure

You can install PCF on Azure either manually or using Terraform.

To install PCF on Azure, do one of the following:

- Install PCF on Azure manually. See Installing PCF on Azure Manually.
- Install PCF on Azure using Terraform. See Installing PCF on Azure Using Terraform.

Install PCF on Azure Government Cloud

To install PCF on Azure Government Cloud, see Configuring BOSH Director on Azure Government Cloud.

Install PCF on Azure Germany

To install PCF in Azure Germany, see Configuring BOSH Director on Azure Germany.

Additional Resources

The following are additional resources related to installing PCF on GCP:

- For information about troubleshooting the PCF on Azure install process, see Troubleshooting PCF on Azure.
- For production-level deployment options for PCF on Azure, see Reference Architecture for Pivotal Cloud Foundry on Azure.
- For information about how to manage IaaS users and credentials in Azure, see Azure security documentation.
- For recommendations on how to create and scope Azure accounts for PCF, see IaaS Permissions Guidelines.
- For information about deleting a PCF on Azure installation, see Deleting a PCF on Azure Installation.
- For information about upgrading PCF on Azure, see Upgrading BOSH Director on Azure.
Installing PCF on Azure Manually

This topic explains how to install Pivotal Cloud Foundry (PCF) on Microsoft Azure manually.

To install PCF on Azure manually, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on Azure Manually.

2. Configure BOSH Director. See Configuring BOSH Director on Azure Manually.

3. Configure PAS. See Configuring PAS.
Preparing to Deploy Ops Manager on Azure Manually

This topic describes how to prepare Azure to deploy Ops Manager. You must deploy Ops Manager to deploy Pivotal Application Service (PAS) or .

Step 1: Install and Configure the Azure CLI

1. Install the Azure CLI 2.0 by following the instructions for your operating system in Install the Azure CLI in the Microsoft Azure documentation.

2. On the command line, set your cloud with a `name` value by running the following command:

   ```bash
   az cloud set --name CLOUD-NAME
   ```

   Where `CLOUD-NAME` is one of the following values. Use the value that corresponds to the Azure environment on which you are installing Ops Manager:
   - Azure: `AzureCloud`
   - Azure China: `AzureChinaCloud` If logging in to `AzureChinaCloud` fails with a `CERT_UNTRUSTED`, use the latest version of node, 4.x or later. For more information about this error, see Failed to login AzureChinaCloud in the Azure/azure-xplat-cli GitHub repository.
   - Azure Government Cloud: `AzureUSGovernment`
   - Azure Germany: `AzureGermanCloud`

   For example:

   ```bash
   $ az cloud set --name AzureCloud
   ```

   **Note:** For more information about installing Ops Manager in the China Region, see Install in the China Region in Installing PCF in Airgapped Environments in the Pivotal documentation.

3. To log in, run the following command:

   ```bash
   az login
   ```

   Authenticate by navigating to the URL in the output from the command, entering the provided code, and clicking your account.

Step 2: Set Your Default Subscription

1. To list your Azure subscriptions, run the following command:

   ```bash
   az account list
   ```

   For example:
2. Identify your default subscription in the output of this command by locating the subscription listed that has `isDefault` set to `true`. To deploy Ops Manager to a different subscription, you must set the other subscription as the default. To set a new default subscription, run the following command:

```bash
az account set --subscription SUBSCRIPTION-ID
```

Where `SUBSCRIPTION-ID` is the value of the `id` field of the subscription that you want to set as the default.

The following example sets the default to a subscription that has an `id` with the value of "87654321-1234-5678-1234-567891234567".

```bash
$ az account set --subscription "87654321-1234-5678-1234-567891234567"
```

3. Record the value of the `id` of the default subscription. You use this value in future configuration steps.

4. Record the value of `tenantId` of the default subscription. This is your `TENANT_ID` for creating a service principal. If your `tenantId` value is not defined, you may be using a personal account to log in to your Azure subscription.

### Step 3: Create an AAD Application

1. To create an Azure Active Directory (AAD) application, run the following command:

```bash
az ad app create --display-name "Service Principal for BOSH" \
--password "PASSWORD" --homepage "http://BOSHAzureCPI" \
--identifier-uris "http://BOSHAzureCPI"
```

Where `PASSWORD` is a password of your choice. This is your `CLIENT_SECRET` for creating a service principal.

For example:

```bash
$ az ad app create --display-name "Service Principal for BOSH" \
--password "Swordfish" --homepage "http://BOSHAzureCPI" \
--identifier-uris "http://BOSHAzureCPI"
```

**Note:** You can provide any string for the `homepage` and `identifier-uris` flags, but the value of `identifier-uris` must be unique within the organization associated with your Azure subscription. For the `homepage`, Pivotal recommends using `http://BOSHAzureCPI` as shown in the example above.

2. From the output of the `az ad app create` that you ran in the previous step, locate and record the value of `appId`. This is your `APPLICATION_ID` for creating a service principal.
Step 4: Create and Configure a Service Principal

1. To create a service principal, run the following command:

   ```bash
   az ad sp create --id YOUR-APPLICATION-ID
   ```

   Where **YOUR-APPLICATION-ID** is the **APPLICATION_ID** that you recorded in Create an AAD Application. For example:

   ```bash
   az ad sp create --id 5c552e8f-b977-45f5-a50b-981cfe17c0b0
   ```

2. You must have the Contributor role on your service principal to deploy Ops Manager to Azure. To assign the Contributor role on your service principal, run the following command:

   ```bash
   az role assignment create --assignee "SERVICE-PRINCIPAL-NAME" --role "Contributor" --scope /subscriptions/SUBSCRIPTION-ID
   ```

   Where:
   - **SERVICE-PRINCIPAL-NAME** is any value of **servicePrincipalNames** from the output above, such as **YOUR-APPLICATION-ID**.
   - **SUBSCRIPTION-ID** is the value of the **id** of the default subscription that you recorded in Set Your Default Subscription. For example:

   ```bash
   az role assignment create --assignee "5c552e8f-b977-45f5-a50b-981cfe17c0b0" --role "Contributor" --scope /subscriptions/87654321-1234-5678-1234-567891234567
   ```

   **Note:** If you need to use multiple resource groups for your deployment on Azure, you can define custom roles for your Service Principal. These roles allow BOSH to deploy to pre-existing network resources outside of the resource group. For more information, see Multiple Resource Group Deployment in Reference Architecture for Pivotal Cloud Foundry on Azure in the Pivotal documentation.

   For more information about Azure Role-Based Access Control, refer to Built-in roles for Azure resource in the Microsoft Azure documentation.

3. To verify the role assignment, run the following command:

   ```bash
   az role assignment list --assignee "SERVICE-PRINCIPAL-NAME"
   ```

   Where **SERVICE-PRINCIPAL-NAME** is any value of **servicePrincipalNames** from the output above, such as **YOUR-APPLICATION-ID**. For example:
Step 5: Verify Your Service Principal

To verify your service principal, run the following command to log in to your service principal:

```
az login --username APPLICATION_ID --password CLIENT_SECRET \n--service-principal --tenant TENANT_ID
```

Where:

- **APPLICATION_ID** is the APPLICATION_ID that you recorded in Create an AAD Application.
- **CLIENT_SECRET** is the password that you provided in Create an AAD Application.
- **TENANT_ID** is the value of tenantID of the default subscription that you recorded in Set Your Default Subscription.

For example:

```
$ az login --username 5c552e8f-b977-45f5-a50b-981cfe17cb9d \n--password "Swordfish" \n--service-principal --tenant 22222222-1234-5678-1234-678912345678
```

If you cannot log in, the service principal is invalid. Create a new service principal and try again.

Step 6: Perform Registrations

1. To register your subscription with Microsoft.Storage, run the following command:

```
az provider register --namespace Microsoft.Storage
```

2. To register your subscription with Microsoft.Network, run the following command:

```
az provider register --namespace Microsoft.Network
```

3. To register your subscription with Microsoft.Compute, run the following command:

```
az provider register --namespace Microsoft.Compute
```
Next Steps

After you complete this procedure, follow the instructions in Deploying Ops Manager on Azure Manually.
Deploying Ops Manager on Azure Manually

This topic describes how to deploy BOSH and Ops Manager for Pivotal Cloud Foundry (PCF) on Azure by using individual commands to create resources. Pivotal recommends this manual procedure for deploying to Azure China, Azure Germany, and Azure Government Cloud.

Before you perform the procedures in this topic, you must have completed the procedures in Preparing to Deploy Ops Manager on Azure Manually. After you complete the procedures in this topic, follow the instructions in the Configuring BOSH Director on Azure topic.

Note: If you are deploying BOSH and Ops Manager on Azure Stack, complete the procedures in Install and configure CLI for use with Azure Stack in the Microsoft documentation before following the procedures in this topic.

Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

Step 1: Create Network Resources

1. Navigate to the Azure portal, click Resource groups, and click Add to create a new resource group for your deployment.

2. Enter a Resource group name, select your Subscription, and select a Resource group location. Click Create.

3. Export the name of your resource group as the environment variable $RESOURCE_GROUP.

   

   

4. Export your location. For example, westus.

   

   

5. Create a network security group named pcf-nsg.

   

6. Add network security group rules to the pcf-nsg group to allow traffic to known ports from the public Internet.

   

   

   

   

   

© Copyright Pivotal Software Inc, 2013-2019
To block traffic from the public Internet, append `--source-address-range AzureLoadBalancer` to allow traffic from only the Azure load balancer or `--source-address-range VirtualNetwork` to only allow traffic from the virtual network.

To allow traffic from both the Azure load balancer and the virtual network, create duplicates of each rule, one specifying `--source-address-range AzureLoadBalancer` and one specifying `--source-address-range VirtualNetwork`.

7. Create a network security group named `opsmgr-nsg`.

    $ az network nsg create --name opsmgr-nsg 
    --resource-group $RESOURCE_GROUP 
    --location LOCATION

8. Add a network security group rule to the `opsmgr-nsg` group to allow HTTP traffic to the Ops Manager VM.

    $ az network nsg rule create --name http 
    --nsg-name opsmgr-nsg 
    --resource-group $RESOURCE_GROUP 
    --protocol Tcp --priority 100 
    --destination-port-range 80

9. Add a network security group rule to the `opsmgr-nsg` group to allow HTTPS traffic to the Ops Manager VM.

    $ az network nsg rule create --name https 
    --nsg-name opsmgr-nsg 
    --resource-group $RESOURCE_GROUP 
    --protocol Tcp --priority 200 
    --destination-port-range 443

10. Add a network security group rule to the `opsmgr-nsg` group to allow SSH traffic to the Ops Manager VM.

    $ az network nsg rule create --name ssh 
    --nsg-name opsmgr-nsg 
    --resource-group $RESOURCE_GROUP 
    --protocol Tcp --priority 300 
    --destination-port-range 22

Instructions: To block traffic from the public Internet, append `--source-address-range AzureLoadBalancer` to allow traffic from only the Azure load balancer or `--source-address-range VirtualNetwork` to only allow traffic from the virtual network. To allow traffic from both the Azure load balancer and the virtual network, create duplicates of each rule, one specifying `--source-address-range AzureLoadBalancer` and one specifying `--source-address-range VirtualNetwork`.

Optionally, if you want to use private IP ranges with Ops Manager and allow all internal traffic, you can create the Network Security Groups to allow all internal traffic.

    $ az network nsg rule create --name internal-internal-network 
    --nsg-name internal-traffic 
    --resource-group $RESOURCE_GROUP 
    --protocol Tcp --priority 100 
    --destination-port-range * 
    --source-address-range VirtualNetwork

    $ az network nsg rule create --name internal-from-internal-network 
    --nsg-name internal-from-internal-network 
    --resource-group $RESOURCE_GROUP 
    --protocol Tcp --priority 110 
    --destination-port-range * 
    --source-address-range AzureLoadBalancer

11. Create a virtual network named `pcf-virtual-network`.

    $ az network vnet create --name pcf-virtual-network 
    --resource-group $RESOURCE_GROUP --location LOCATION 
    --address-prefixes 10.0.0.0/16

12. Add subnets to the network for Ops Manager, BOSH director, and the VMs for your runtime, and attach the Network Security Group.
Step 2: Create BOSH and Deployment Storage Accounts

PCF on Azure uses multiple general-purpose Azure storage accounts. The BOSH and Ops Manager VMs use one main BOSH account, and the other components share five or more deployment storage accounts.

1. Choose a name for your BOSH storage account, and export it as the environment variable \$STORAGE_NAME. Storage account names must be globally unique across Azure, between 3 and 24 characters in length, and contain only lowercase letters and numbers.

\$ export STORAGE_NAME="YOUR-BOSH-STORAGE-ACCOUNT-NAME"

2. Create a Standard storage account for BOSH with the following command. This account will be used for BOSH bookkeeping and running the Ops Manager VM itself, but does not have to be used for running any other VMs.

\$ az storage account create --name $STORAGE_NAME
  --resource-group $RESOURCE_GROUP
  --sku Standard_LRS
  --location $LOCATION

If the command fails, ensure you have followed the rules for naming your storage account. Export another new storage account name if necessary.

3. Configure the Azure CLI to use the BOSH storage account as its default.

   a. Retrieve the connection string for the account.

   \$ az storage account show-connection-string
     --name $STORAGE_NAME
     --resource-group $RESOURCE_GROUP

   The command returns output similar to the following:

   ```
   
   "connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=cfdocsboshstorage;AccountKey=EXAMPLEaaaaabbbrnc5igFxYWsgq016Tu9uGwseOl8bqNBEL/2tp7wX92QMUM19Pz9BYTXt8aq4A=="
   ```

   b. Record the full value of \/connectionString\ from the output above, starting with and including \DefaultEndpointsProtocol=\.

   c. Export the value of \connectionString\ as the environment variable \$CONNECTION_STRING.

   \$ export CONNECTION_STRING="YOUR-ACCOUNT-KEY-STRING"

4. Create three blob containers in the BOSH storage account, named \opsmanager\, \bosh\, and \stemcell\.

\$ az storage container create --name opsmanager
  --connection-string \$CONNECTION_STRING

\$ az storage container create --name bosh
  --connection-string \$CONNECTION_STRING

\$ az storage container create --name stemcell
  --public-access blob
  --connection-string \$CONNECTION_STRING

5. Create a table named \stemcells\.

Note: \Standard_LRS\ refers to a Standard Azure storage account. The BOSH Director requires table storage to store stemcell information. Azure Premium storage does not support table storage and cannot be used for the BOSH storage account.
6. Choose a set of unique names for five or more deployment storage accounts. As with the BOSH storage account above, the names must be unique, alphanumeric, lowercase, and 3-24 characters long. The account names must also be sequential or otherwise identical except for the last character. For example: `xyzdeploymentstorage1`, `xyzdeploymentstorage2`, `xyzdeploymentstorage3`, `xyzdeploymentstorage4`, and `xyzdeploymentstorage5`.

7. Decide which type of storage to use and run the corresponding command below:

   ![Note: Pivotal recommends five Premium storage accounts, which provides a reasonable amount of initial storage capacity. You can use either Premium or Standard storage accounts, but they have very different scalability metrics. Pivotal recommends creating 1 Standard storage account for every 30 VMs, or 1 Premium storage account for every 150 VMs. You can increase the number of storage accounts later by provisioning more and following the naming sequence.]

   - To use Premium storage (recommended):
     ```bash
     az export STORAGE_TYPE="Premium_LRS"
     ```
   - To use Standard storage:
     ```bash
     az export STORAGE_TYPE="Standard_LRS"
     ```

8. For each deployment storage account, do the following:

   a. Create the storage account with the following command, replacing `MY_DEPLOYMENT_STORAGE_X` with one of your deployment storage account names.

     ```bash
     az storage account create --name MY_DEPLOYMENT_STORAGE_X --resource-group $RESOURCE_GROUP --sku $STORAGE_TYPE --kind Storage --location $LOCATION
     ```

     If the command fails, try a different set of account names.

   b. Retrieve the connection string for the account.

     ```bash
     az storage account show-connection-string --name MY_DEPLOYMENT_STORAGE_X --resource-group $RESOURCE_GROUP
     ```

     The command returns output similar to the following:

     ```json
     {
     "connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=cfdocsdeploystorage1;AccountKey=EXAMPLEaaaaaaaQiSAmqj1OocsGhKBwnMf8wEwdeJMvvonrbmNk27bfkSL8ZFzAhs3Kb78si5CTPHhjHHiK4qPcYzn/8OmFg=="
     }
     ```

   c. Record the full value of `connectionString` from the output above, starting with and including `DefaultEndpointsProtocol=`,

   d. Create two blob containers named `bosh` and `stemcell` in the account.

      ```bash
      az storage container create --name bosh --connection-string $CONNECTION_STRING
      az storage container create --name stemcell --connection-string $CONNECTION_STRING
      ```

**Step 3: Create Load Balancers**

Your load balancer configuration depends on whether you want apps to be available from public IP addresses, private IP addresses, or both.

1. **Required: PAS Load Balancer**

   a. Create a load balancer named `pcf-lb`. A static IP address will be automatically created for Standard SKU load balancers unless specified otherwise with `--public-ip-address-allocation Dynamic`.  

© Copyright Pivotal Software Inc, 2013-2019 192 2.5
5 az network lb create --name pcf-lb \ 
--resource-group $RESOURCE_GROUP --location $LOCATION \ 
--backend-pool-name pcf-lb-be-pool --frontend-ip-name pcf-lb-fe-ip \ 
--public-ip-address pcf-lb-ip --public-ip-address-allocation Static \ 
--sku Standard

**Note:** If the Standard SKU is not available in Azure China, you can change to use the Basic SKU.

**Note:** This back end pool is empty when you create it.

b. Add a probe to the load balancer.

5 az network lb probe create --lb-name pcf-lb \ 
--name http8080 --resource-group $RESOURCE_GROUP \ 
--protocol Http --port 8080 --path health

c. Add a load balancing rule for HTTP.

5 az network lb rule create --lb-name pcf-lb \ 
--name http --resource-group $RESOURCE_GROUP \ 
--protocol Tcp --frontend-port 80 --backend-port 80 --frontend-ip-name pcf-lb-fe-ip \ 
--backend-pool-name pcf-lb-be-pool --probe-name http8080

d. Add a load balancing rule for HTTPS.

5 az network lb rule create --lb-name pcf-lb \ 
--name https --resource-group $RESOURCE_GROUP \ 
--protocol Tcp --frontend-port 443 --backend-port 443 --frontend-ip-name pcf-lb-fe-ip \ 
--backend-pool-name pcf-lb-be-pool --probe-name http8080

e. (Optional) For private IPs, do the following:

- Create a load balancer named pcf-lb.

5 az network lb create --name pcf-lb \ 
--resource-group $RESOURCE_GROUP --location $LOCATION \ 
--backend-pool-name pcf-lb-be-pool --frontend-ip-name pcf-lb-fe-ip \ 
--private-ip-address 10.0.0.6 --sku Standard

- Add a probe to the load balancer.

5 az network lb probe create --lb-name pcf-lb \ 
--name http8080 --resource-group $RESOURCE_GROUP \ 
--protocol Http --port 8080

- Add a load balancing rule for HTTP.

5 az network lb rule create --lb-name pcf-lb \ 
--name http --resource-group $RESOURCE_GROUP \ 
--protocol Tcp --frontend-port 80 --backend-port 80 --frontend-ip-name pcf-lb-fe-ip \ 
--backend-pool-name pcf-lb-be-pool --probe-name http8080

- Add a load balancing rule for HTTPS.

5 az network lb rule create --lb-name pcf-lb \ 
--name https --resource-group $RESOURCE_GROUP \ 
--protocol Tcp --frontend-port 443 --backend-port 443 --frontend-ip-name pcf-lb-fe-ip \ 
--backend-pool-name pcf-lb-be-pool --probe-name http8080

**Note:** If the Standard SKU is not available in Azure China, you can change to use the Basic SKU.
Navigate to your DNS provider and create an entry that points \texttt{*.YOUR-SUBDOMAIN} to the public IP address of your load balancer. For example, create an entry that points \texttt{azure.example.com} to \texttt{198.51.100.1}.

\textbf{Note:} You can retrieve it by \texttt{az network public-ip show --name pcf-lb-ip --resource-group \$RESOURCE_GROUP}.

2. \textbf{Optional:} Diego SSH Load Balancer
   
   a. Create a load balancer named \texttt{pcf-ssh-lb}.

   ```bash
   $ az network lb create --name pcf-ssh-lb
   --resource-group \$RESOURCE_GROUP --location \$LOCATION
   --backend-pool-name pcf-ssh-lb-be-pool --frontend-ip-name pcf-ssh-lb-fe-ip
   --public-ip-address pcf-ssh-lb-ip --public-ip-address-allocation Static
   --sku Standard
   ```

   \textbf{Note:} If the Standard SKU is not available in Azure China, you can change to use the Basic SKU.

   \textbf{Note:} This back end pool is empty when you create it.

   b. (Optional) For private IPs, create a load balancer named \texttt{pcf-ssh-lb}.

   ```bash
   $ az network lb create --name pcf-ssh-lb
   --resource-group \$RESOURCE_GROUP --location \$LOCATION
   --backend-pool-name pcf-ssh-lb-be-pool --frontend-ip-name pcf-ssh-lb-fe-ip
   --private-ip-address 10.0.0.7
   --sku Standard
   ```

   \textbf{Note:} If the Standard SKU is not available in Azure China, you can change to use the Basic SKU.

   \textbf{Note:} This back end pool is empty when you create it.

   c. Add a probe to the load balancer.

   ```bash
   $ az network lb probe create --lb-name pcf-ssh-lb
   --name tcp2222 --resource-group \$RESOURCE_GROUP
   --protocol Tcp --port 2222
   ```

   d. Add a load balancing rule for SSH.

   ```bash
   $ az network lb rule create --lb-name pcf-ssh-lb
   --name diego-ssh --resource-group \$RESOURCE_GROUP
   --protocol Tcp --frontend-port 2222
   --backend-port 2222
   --frontend-ip-name pcf-ssh-lb-fe-ip
   --backend-pool-name pcf-ssh-lb-be-pool
   --probe-name tcp2222
   ```

e. Navigate to your DNS provider, and create an entry that points \texttt{ssh.sys.YOUR-SUBDOMAIN} to the public IP address of your load balancer. For example, create an entry that points \texttt{azure.example.com} to \texttt{198.51.100.1}.

   \textbf{Note:} You can retrieve it by \texttt{az network public-ip show --name pcf-ssh-lb-ip --resource-group \$RESOURCE_GROUP}.

Step 4: Boot Ops Manager

1. Navigate to Pivotal Network and download the latest release of Pivotal Cloud Foundry Ops Manager for Azure.

2. View the downloaded PDF and locate the Ops Manager image URL appropriate for your region.

3. Export the Ops Manager image URL as an environment variable.
4. Download the Ops Manager image. For compatibility when upgrading to future versions of Ops Manager, choose a unique name for the image that includes the Ops Manager version number. For example, replace `opsman-image-2.3.x` in the following examples with `opsman-image-2.3.1`.

   - If you use unmanaged disks, perform the following steps:

     ```sh
     $ export OPS_MAN_IMAGE_URL="YOUR-OPS-MAN-IMAGE-URL"
     
     4. Download the Ops Manager image. For compatibility when upgrading to future versions of Ops Manager, choose a unique name for the image that includes the Ops Manager version number. For example, replace `opsman-image-2.3.x` in the following examples with `opsman-image-2.3.1`.
     
     ```sh
     1. If you use unmanaged disks, perform the following steps:
     
        ```sh
        Note: Azure Stack requires unmanaged disks.
        
        1. Download the Ops Manager image to your local machine. The image size is 10 GB.
           
           ```sh
           $ wget $OPS_MAN_IMAGE_URL -O opsman-image-2.3.x.vhd
           
           2. Upload the image to your storage account using the Azure CLI.
           
           ```sh
           $ az storage blob upload --name opsman-image-2.3.x.vhd 
           --connection-string $CONNECTION_STRING 
           --container-name opsmanager 
           --type page 
           --file opsman-image-2.3.x.vhd
           
           ```sh
           2. If you use managed disks, do the following:
           
           ```sh
           Note: If you do not want to use a public IP for Ops Manager, skip this step.
           ```sh
           ```sh
           1. Copy the Ops Manager image into your storage account using the Azure CLI.
           
           ```sh
           $ az storage blob copy start --source-uri $OPS_MAN_IMAGE_URL 
           --connection-string $CONNECTION_STRING 
           --destination-container opsmanager 
           --destination-blob opsman-image-2.3.x.vhd
           
           2. Copying the image may take several minutes. Run the following command and examine the output under "copy".
           
           ```sh
           $ az storage blob show --name opsman-image-2.3.x.vhd 
           --container-name opsmanager 
           --connection-string $CONNECTION_STRING
           ```sh
           ```sh
           3. Wait a few moments and re-run the command above if status is pending. When status reads success, continue to the next step.
           ```sh
           ```sh
           5. Create a public IP address named ops-manager-ip.
           
           ```sh
           $ az network public-ip create --name ops-manager-ip 
           --resource-group $RESOURCE_GROUP --location LOCATION 
           --allocation-method Static
           ```sh
           ```sh
           Note: If you do not want to use a public IP for Ops Manager, skip this step.
6. Record the *ipAddress* from the output above. This is the public IP address of Ops Manager.

7. Create a network interface for Ops Manager.

```bash
az network nic create --name pcf-nic --vnet-name pcf-virtual-network --subnet pcf-infrastructure-subnet --network-security-group pcf-nsg-manager-ip --resource-group $RESOURCE_GROUP --private-ip-address 10.0.4.4 --public-ip-address ops-manager-ip --resource-group $RESOURCE_GROUP --name opsman-nic --location $LOCATION
```

**Note:** If you do not want to use a public IP address for Ops Manager, remove the `--public-ip-address ops-manager-ip` flag and value.

8. Create a keypair on your local machine with the username *ubuntu*. For example, enter the following command:

```bash
ssh-keygen -t rsa -f opsman -C ubuntu
```

When prompted for a passphrase, press the *enter* key to provide an empty passphrase.

9. Create the Ops Manager VM.

- If you are using unmanaged disks, run the following command to create your Ops Manager VM, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key .pub file:

```bash
az vm create --name opsman-2.3.x --resource-group $RESOURCE_GROUP --location $LOCATION --nics opsman-nic --image https://$STORAGE_NAME.my-azure-instance.com/opsmanager/opsman-image-2.3.x.vhd --os-disk-name opsman-2.3.x-osdisk --os-disk-size-gb 128 --os-type Linux --storage-account $STORAGE_NAME --storage-container-name opsmanager --admin-username ubuntu --ssh-key-value PATH-TO-PUBLIC-KEY
```

Replace `my-azure-instance.com` with the URL of your Azure instance. Find the complete source URL in the Azure UI by viewing the Blob properties of the Ops Manager image you created earlier in this procedure.

- If you are using Azure managed disks, do the following:

  1. Create a managed image from the Ops Manager VHD file:

```bash
az image create --resource-group $RESOURCE_GROUP --name opsman-image-2.3.x --source https://$STORAGE_NAME.blob.core.windows.net/opsmanager/image-2.3.x.vhd --location $LOCATION --os-type Linux
```

If you are using Azure China, Azure Government Cloud, or Azure Germany, replace `blob.core.windows.net` with the following:

- For Azure China, use `blob.core.chinacloudapi.cn`. See the Azure documentation for more information.
- For Azure Government Cloud, use `blob.core.usgovcloudapi.net`. See the Azure documentation for more information.
- For Azure Germany, use `blob.core.cloudapi.de`. See the Azure documentation for more information.

  2. Create your Ops Manager VM, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key .pub file.

```bash
az vm create --name opsman-2.3.x --resource-group $RESOURCE_GROUP --location SLOCATION --nics opsman-nic --image opsman-image-2.3.x --os-disk-size-gb 128 --os-disk-name opsman-2.3.x-osdisk --admin-username ubuntu --size Standard_DS2_v2 --storage-sku Standard_LRS --ssh-key-value PATH-TO-PUBLIC-KEY
```
If you plan to install more than one tile in this Ops Manager installation, do the following to increase the size of the Ops Manager VM disk. You can repeat this process and increase the disk again at a later time if necessary.

**Note:** If you use Azure Stack, you must increase the Ops Manager VM disk size using the Azure Stack UI.

a. Run the following command to stop the VM and detach the disk:

```
$ az vm deallocate --name opsman-2.3.x \
--resource-group $RESOURCE_GROUP
```

b. Run the following command to resize the disk to 128 GB:

```
$ az disk update --size-gb 128 --name opsman-2.3.x-osdisk \
--resource-group $RESOURCE_GROUP
```

c. Run the following command to start the VM:

```
$ az vm start --name opsman-2.3.x --resource-group $RESOURCE_GROUP
```

Step 5: Complete BOSH Director Configuration

1. Navigate to your DNS provider, and create an entry that points a fully qualified domain name (FQDN) to the public IP address of Ops Manager. As a best practice, always use the FQDN to access Ops Manager.

2. Continue to the Configuring BOSH Director on Azure Manually topic.
Configuring BOSH Director on Azure Manually

This topic describes how to configure the BOSH Director tile within Ops Manager on Azure after Deploying Ops Manager on Azure Manually.

Prerequisites

See the following sections to prepare for configuring BOSH Director on Azure.

General Prerequisites

Before you perform the procedures in this topic, you must have completed the procedures in the following topics:

- Preparing to Deploy Ops Manager on Azure Manually
- Deploying Ops Manager on Azure Manually

Note: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Prerequisites for Azure Stack or Azure Government Cloud

If you use Azure Government Cloud, you must set custom VM types for a successful deployment. You can set custom VM types with the Ops Manager API.

For more information, see Using the Ops Manager API.

Example curl command to set a VM types:

```bash
  { "name": "Standard_D1_v2", "ram": 3584, "cpu": 1, "ephemeral_disk": 51200 },
  { "name": "Standard_D2_v2", "ram": 7168, "cpu": 2, "ephemeral_disk": 102400 },
  { "name": "Standard_D3_v2", "ram": 14336, "cpu": 4, "ephemeral_disk": 204800 },
  { "name": "Standard_D4_v2", "ram": 28672, "cpu": 8, "ephemeral_disk": 409600 },
  { "name": "Standard_D5_v2", "ram": 57344, "cpu": 8, "ephemeral_disk": 819200 },
  { "name": "Standard_D11_v2", "ram": 14336, "cpu": 2, "ephemeral_disk": 102400 },
  { "name": "Standard_D12_v2", "ram": 28672, "cpu": 4, "ephemeral_disk": 204800 },
  { "name": "Standard_D13_v2", "ram": 57344, "cpu": 8, "ephemeral_disk": 409600 },
  { "name": "Standard_D14_v2", "ram": 114688, "cpu": 16, "ephemeral_disk": 819200 },
  { "name": "Standard_F1", "ram": 2048, "cpu": 1, "ephemeral_disk": 16384 },
  { "name": "Standard_F2", "ram": 4096, "cpu": 2, "ephemeral_disk": 32768 },
  { "name": "Standard_F4", "ram": 8192, "cpu": 4, "ephemeral_disk": 65536 },
  { "name": "Standard_F8", "ram": 16384, "cpu": 8, "ephemeral_disk": 131072 }]
]'
```

To verify that you have successfully set your custom VM types, use the GET /api/v0/vm_types Ops Manager API endpoint.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain name (FQDN) of Ops Manager that you created in Deploying Ops Manager on Azure Manually.

2. When Ops Manager starts for the first time, you must choose one of the following:

- **Internal Authentication**: If you use Internal Authentication, Ops Manager maintains your user database.
- **SAML Identity Provider**: If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
- **LDAP Server**: If you use a LDAP Server, an external identity server maintains your user database.
Internal Authentication

1. When redirected to the Internal Authentication page, do the following:
   - Enter a Username, Password, and Password confirmation to create an Admin user.
   - Enter a Decryption passphrase and the Decryption passphrase confirmation. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an HTTP proxy or HTTPS proxy, follow the instructions in Configuring Proxy Settings for the BOSH CPI.
   - Read the End User License Agreement, and select the checkbox to accept the terms.
   - Click Setup Authentication.

2. Log in to Ops Manager with the Admin username and password you created in the previous step.
SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.

3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your Decryption passphrase. Read the End User License Agreement, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click Login.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - 6a. Ops Manager SAML service provider metadata: https://OPS-MAN-FQDN:443/uaa/saml/metadata
   - 6b. BOSH Director SAML service provider metadata: https://BOSH-IP-ADDRESS:8443/saml/metadata

   **Note:** To retrieve your BOSH-IP-ADDRESS, navigate to the Status tab in the BOSH Director tile. Record the BOSH Director IP address.
7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.

- **Audience URI (SP Entity ID)**: [https://OP-MAN-FQDN:443/uaa](https://OP-MAN-FQDN:443/uaa)
- **Name ID**: Email Address
- **SAML authentication requests are always signed**

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.

- **Single sign on URL**: [https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP](https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP)
- **Audience URI (SP Entity ID)**: [https://BOSH-IP:8443](https://BOSH-IP:8443)
- **Name ID**: Email Address
- **SAML authentication requests are always signed**

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

### LDAP Server

- **For Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - **ldap://**: This specifies that the LDAP server uses an unencrypted connection.
  - **ldaps://**: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or that you import a trusted certificate to the JVM truststore.

- **For LDAP Username and LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: `cn=adminstrator,ou=Users,dc=example,dc=com`

  **Note**: Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

  **warning**: Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- **For User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`

- **For User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

  In the LDAP search filter string that you use to configure your runtime, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

  In addition to `cn`, other attributes commonly searched for and returned are `mail`, `uid`, and, in the case of Active Directory, `sAMAccountName`.

  **Note**: For instructions for testing and troubleshooting your LDAP search filters, see [Configuring LDAP Integration with Pivotal Cloud Foundry](#).

---

© Copyright Pivotal Software Inc, 2013-2019
For Group Search Base, enter the location in the LDAP directory tree from which the LDAP Group search begins.

For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base:

```
ou=Groups,dc=example,dc=com
```

For Group Search Filter, enter a string that defines LDAP Group search criteria. The standard value is `member={0}`.

For Email Attribute, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example `mail`.

For LDAP RBAC Admin Group Name, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.

From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:

- Automatically follow any referrals.
- Ignore referrals and return partial result.
- Throw exception for each referral and abort.

For Server SSL Cert, paste in the root certificate from your CA certificate or your self-signed certificate.

Enter a Decryption passphrase and the Decryption passphrase confirmation. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

If you are using an HTTP proxy or HTTPS proxy, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

Read the End User License Agreement, and select the checkbox to accept the terms.

Select Provision an admin client in the BOSH UAA. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

Click Setup Authentication.

Return to the BOSH Director tile, and continue with the configuration steps below.

---

Step 2: Azure Config Page

1. Click the BOSH Director tile.

   ![BOSH Director for Azure](image)

2. Select Azure Config.
3. Complete the following fields with information you obtained in Preparing to Deploy Ops Manager on Azure Manually.
- **Subscription ID**: Enter the ID of your Azure subscription.
- **Tenant ID**: Enter your TENANT_ID.
- **Application ID**: Enter the APPLICATION_ID that you created in the Create an AAD Application step of Preparing to Deploy Ops Manager on Azure Manually.
- **Client Secret**: Enter your CLIENT_SECRET.

4. Complete the following fields:
   - **Resource Group Name**: Enter the name of your resource group, which you exported as the $RESOURCE_GROUP environment variable.
   - **BOSH Storage Account Name**: Enter the name of your storage account, which you exported as the $STORAGE_NAME environment variable.

5. For **Cloud Storage Type**, select one of the following options based on your Azure VM storage settings.
   - **Use Managed Disks**: Select this option if you use Azure Managed Disks. See Azure Managed Disks Overview in the Microsoft documentation for more information. For **Storage Account Type**, select the storage option that corresponds with your Azure subscription. Select Standard_LRS for HDD-based storage or Premium_LRS for SSD-based storage.
   - **Use Storage Accounts**: Select this option if you use storage accounts to store your Azure VMs. Enter the base storage name that you used to create your deployment storage accounts, prepended and appended with the wildcard character *. For example, if you created accounts named xyzdeploymentstorage1, xyzdeploymentstorage2, and xyzdeploymentstorage3, enter *deploymentstorage*. Ops Manager requires that you specify an asterisk at both the beginning and the end of the base storage account name.

6. (Optional) Enter your **Default Security Group**. For more information about Network Security Groups, see Filter network traffic with network security groups in Microsoft Azure's documentation.

   ![Use Storage Accounts](image1)

   **Note**: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

7. For **SSH Public Key**, copy and paste the contents of your public key in the opsman.pub file. You created this file in Deploying Ops Manager on Azure Manually.

8. For **SSH Private Key**, copy and paste the contents of your private key in the opsman file.

9. For **Availability Mode**, choose whether to use Availability Sets or Availability Zones. You can further configure this option in the Assign AZs and Networks pane. For more information, see Assign AZs and Networks.
   - For **new installations of Ops Manager v2.5**: Availability Zones are selected by default, but you can choose Availability Sets.
   - If you are upgrading from v2.4: You must use Availability Sets.

   ![Note](image2)


   ![Note](image3)

---

© Copyright Pivotal Software Inc, 2013-2019
11. (Optional) If you selected Azure Stack for your Azure Environment, complete the following Azure Stack-only fields:
   a. For Domain, enter the domain for your Azure Stack deployment. For example, local.azurestack.external.
   b. Enter the Tenant Management Resource Endpoint from your AzureRM Environment Context. This URL changes with every ASDK installation.
   c. Enter AzureAD for Authentication. If you want to use Azure China Cloud Active Directory for your authentication type, enter AzureChinaCloudAD.
   d. Enter your Azure Stack Endpoint Prefix. For example, management.
   e. Enter your Azure Stack CA Certificate. Azure Stack requires a custom CA certificate. Copy the certificate from your Azure Stack environment.

12. Click Save.

Step 3: Director Config Page

1. In Ops Manager, select Director Config.
   ![Director Config](image)

2. In the NTP Servers (comma delimited) field, enter a comma-separated list of valid NTP servers.

   Note: The NTP server configuration only updates after VM recreation. Ensure that you select the Recreate all VMs checkbox if you modify the value of this field.

3. Leave the JMX Provider IP Address field blank.

4. Leave the Bosh HM Forwarder IP Address field blank.

   Note: Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the Bosh HM Forwarder IP Address field blank.

5. Select the Enable VM Resurrection Plugin checkbox to enable the BOSH Resurrection functionality and increase your runtime availability.

6. Select Enable Post Deploy Scripts to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.
7. Select **Recreate VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. (Optional) Disable **Allow Legacy Agents** if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.

13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


    ![HM Pager Duty Plugin](image)

    - **Service Key**: Enter your API service key from PagerDuty.
    - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

    ![HM Email Plugin](image)

    - **Host**: smtp.example.com
    - **Port**: 25
    - **Domain**: cloudfoundry.example.com
    - **From**: user2@example.com
    - **Username**: user
    - **Password**: 
    - **Enable TLS**
- **Host**: Enter your email hostname.
- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in [Preparing CredHub HSMs for Configuration](#).

**Note**: After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
- **Internal**: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.
- **Luna HSM**: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in [Preparing CredHub HSMs for Configuration](#).
3. **Provider Partition Password**
4. **Provider Client Certificate**: The certificate that validates the identity of the HSM when CredHub connects as a client.
5. **Provider Client Certificate Private Key**
6. **HSM Host Address**
7. **HSM Port Address**: If you do not know your port address, enter 1792.
8. **Partition Serial Number**
9. **HSM Certificate**: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. Select a **Blobstore Location** to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscaleable and less secure, Pivotal recommends that you configure an external blobstore.
- **Internal**: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.

- **Enable TLS**: Select this checkbox to enable TLS to the blobstore.

  **Note**: If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

- **S3 Compatible Blobstore**: Select this option to use an external S3-compatible endpoint. Follow the procedures in [Sign up for Amazon S3](#) and [Creating a Bucket](#) in the AWS documentation. When you have created an S3 bucket, complete the following steps:

  1. **S3 Endpoint**: Navigate to the Regions and Endpoints topic in the AWS documentation.

     a. Locate the endpoint for your region in the Amazon Simple Storage Service (S3) table and construct a URL using your region’s endpoint. For example, if you are using the `us-west-2` region, the URL you create would be `https://s3-us-west-2.amazonaws.com`. Enter this URL into the S3 Endpoint field.

     b. On a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.
c. Copy the custom public CA certificate you used to sign the S3 endpoint into `/etc/ssl/certs` on the Ops Manager VM.

d. On the Ops Manager VM, run `sudo update-ca-certificates -f -v` to import the custom CA certificate into the Ops Manager VM truststore.

> **Note:** You must also add this custom CA certificate to the Trusted Certificates field in the Security page. See Security Page for instructions.

2. **Bucket Name:** Enter the name of the S3 bucket.

3. **Access Key** and **Secret Key:** Enter the keys you generated when creating your S3 bucket.

4. **Select V2 Signature** or **V4 Signature.** If you select **V4 Signature,** enter your **Region.**

> **Note:** AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see [Authenticating Requests](https://docs.aws.amazon.com/AmazonS3/latest/userguide/authenticating-requests.html) in the AWS documentation.

- **GCS Blobstore:** Select this option to use an external GCS endpoint. To create a GCS bucket, you must have a GCS account. Follow the procedures in [Creating Storage Buckets](https://cloud.google.com/storage/docs莆们Creating-a-storage-bucket) in the GCS documentation to create a GCS bucket. When you have created a GCS bucket, complete the following steps:

  1. **Bucket Name:** Enter the name of your GCS bucket.
  2. **Storage Class:** Select the storage class for your GCS bucket. See [Storage Classes](https://cloud.google.com/storage/docs莆们storage-classes) in the GCP documentation for more information.
  3. **Service Account Key:** Follow the steps in the **Set up IAM Service Accounts** section of Preparing to Deploy Ops Manager on GCP Manually to download a JSON file with a private key. Enter the contents of the JSON file into the field.

18. Select a **Database Location.** By default, Ops Manager deploys and manages an **Internal** database for you. If you choose to use an **External MySQL Database,** complete the associated fields with information obtained from your external MySQL Database provider: **Host,** **Port,** **Username,** **Password,** and **Database.**

   ![Database Location](https://example.com)

   In addition, if you selected the **Enable TLS for Director Database** checkbox, you can complete the following optional fields:

   - **Enable TLS:** Select this checkbox enables TLS communication between the BOSH Director and the database.
   - **TLS CA:** Enter the Certificate Authority for the TLS Certificate.
   - **TLS Certificate:** Enter the client certificate for mutual TLS connections to the database.
   - **TLS Private Key:** Enter the client private key for mutual TLS connections to the database.
   - **Advanced DB Connection Options:** If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.

19. (Optional) Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to **5.**

20. (Optional) **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. (Optional) To add a custom URL for your BOSH Director, enter a valid hostname in **Director Hostname.** You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see [How to Set Up a Load Balancer in Front of Operations Manager Director](https://docs.pivotal.io) in the Pivotal Support Knowledge Base.
22. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the Custom SSH Banner field.

23. (Optional) Enter your comma-separated custom Identification Tags. For example, iaa:audit1, hello:world. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

   Note: Azure has limited space for identification tags. Pivotal recommends entering no more than five tags.

24. Click Save.

Step 4: Create Networks Page

Create the following three networks:

- Infrastructure Network
- Runtime Network
- Services Network

Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

Infrastructure Network

Follow the steps below to create the infrastructure network.

1. Select Create Networks.

2. (Optional) Select Enable ICMP checks if you want to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. Click Add Network. For name, enter infrastructure.

4. Under Subnets, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Azure Network Name</th>
<th>CIDR</th>
<th>Reserved IP Ranges</th>
<th>DNS</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>pcf-virtual-network/pcf-infrastructure-subnet</code></td>
<td><code>10.0.4.0/26</code></td>
<td><code>10.0.4.1-10.0.4.9</code></td>
<td><code>168.63.129.16</code></td>
<td></td>
</tr>
</tbody>
</table>
Enter the first IP address of the subnet. For example: 10.0.4.1.

5. Click Save.

**Note:** After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see Expanding Your Network with Additional Subnets.

Runtime Network

Follow the steps below to create the runtime network.

1. Click **Add Network**. For **Name**, enter the name of the runtime that you intend to deploy in this environment. For example, `pas` or `pks`.

2. Under **Subnets**, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Azure Network Name</th>
<th>CIDR</th>
<th>Reserved IP</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pas</code> or <code>pks</code></td>
<td><code>pcf-virtual-network/pcf-infra-subnet</code></td>
<td><code>10.0.4.0/26</code></td>
<td><code>10.0.12.1-10.0.12.9</code></td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019 212 2.5
3. Click Save.

Services Network

Follow the steps below to create the services network.

1. Click Add Network. For Name, enter services.

2. Under Subnets, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Azure Network Name</th>
<th>CIDR</th>
<th>Reserved IP Ranges</th>
<th>DNS</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>services</td>
<td>pcf-virtual-networkpcf-services-subnet</td>
<td>10.0.8.0/22</td>
<td>10.0.8.1-10.0.8.9</td>
<td>168.63.129.16</td>
<td>10.0.8.1</td>
</tr>
</tbody>
</table>

Enter [168.63.129.16].

Enter the first IP address of the subnet, for example, [168.63.129.16].
3. Click **Save**. If **Enable ICMP checks** is not selected, you may see red warnings. You can ignore these warnings.

### Step 5: Assign AZs and Networks

1. Select **Assign AZs and Networks**.

   **Note:** The appearance of this page changes based on whether you are using Availability Zones or Availability Sets. For more information, see [Azure Config Page](#).

   - If you use Availability Zones, select one of the three available zones from the **Singleton Availability Zone** dropdown, then select a network from the **Networks** dropdown. The networks that appear in this list are networks that you created earlier in this procedure.
   - If you use Availability Sets, select a network from the **Networks** dropdown. The networks that appear in this list are networks that you created earlier in this procedure.

2. Click **Save**.

### Step 6: Security Page

1. Select **Security**.
2. In **Trusted Certificates**, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCD...;
------END CERTIFICATE------
-----BEGIN CERTIFICATE-----
CDEFGH12345678ABCDEFGH12345678ABCDEF
EFGHI2345678ABCD...;
------END CERTIFICATE------
```

3. (Optional) Select the **Include OpsManager Root CA in Trusted Certs** checkbox to include the Ops Manager root CA in the Trusted Certificates field. BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

5. Click **Save**. To view your saved Director password, click the **Credentials** tab.

---

**Step 7: BOSH DNS Config Page**

Note: If you want to use Docker registries to run PAS app instances in Docker containers, enter the certificate for your private Docker registry in this field. See Using Docker Registries for more information about running app instances in PAS using Docker registries.
1. Select BOSH DNS Config.

2. (Optional) In Excluded Recursors, enter a list of prohibited recursor addresses.

3. (Optional) In Recursor Timeout, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   Note: This time limit must include one of the Go parse duration time units. For example, entering `5s` sets the timeout limit to five seconds. For more information about supported time units, see `func ParseDuration` in the Go Programming Language documentation.

4. (Optional) In Handlers, enter a list of custom domain handlers in JSON format.

5. Click Save.

Step 8: Syslog Page
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.
7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for **rsyslog** in the **Custom rsyslog Configuration** field. This field requires the rainerscript syntax.

10. Click **Save Syslog Settings**.

### Step 9: Resource Config Page

1. Select **Resource Config**.

   ![Resource Config](image)

2. Ensure that the **Internet Connected** checkboxes are deselected for all jobs.

3. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the dropdown to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

   **Note:** Ops Manager requires a Director VM with at least 8 GB memory.

   **Note:** If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

   **Note:** If you install PAS for Windows, provision your **Master Compilation Job** with at least 100 GB of disk space.

4. (Optional) For **Load Balancers**, enter your Azure application gateway name for each associated job. To create an application gateway, follow the procedures in [Configure an application gateway for SSL offload by using Azure Resource Manager](#) from the Azure documentation. When you create the application gateway, associate the gateway’s IP address with your system domain.

   **Warning:** This feature is not recommended for production use. The Azure load balancer does not support an override port in the healthcheck configuration.

5. Click **Save**.

### Step 10: (Optional) Add Custom VM Extensions

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see [Managing Custom VM Extensions](#).

### Step 11: Complete the BOSH Director Installation

© Copyright Pivotal Software Inc, 2013-2019
1. Click **Review Pending Changes**, then **Apply Changes**. If a red ICMP error message appears and you have disabled ICMP, click **Ignore errors and start the install**.

2. BOSH Director installs. This may take a few moments. When the installation process successfully completes, the **Changes Applied** window appears.

3. Click the **Installation Dashboard** link to return to the Installation Dashboard.

**Next Steps**

- (Optional) When Ops Manager finishes deploying, deploy BOSH Add-ons to your system. For more information, see [Deploying BOSH Add-Ons](#).
- Install one or more PCF runtime environments to complete your installation. For more information, see [Installing Runtimes](#).
Installing PCF on Azure Using Terraform

This topic explains how to install Pivotal Cloud Foundry (PCF) on Microsoft Azure using Terraform.

To install PCF on Azure using Terraform, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on Azure Using Terraform.
2. Configure BOSH Director. See Configuring BOSH Director on Azure Using Terraform.
3. Configure PAS. See Configuring PAS.
Deploying Ops Manager on Azure Using Terraform

Page last updated:

This guide describes the preparation steps required to deploy Ops Manager to Azure using Terraform templates.

The Azure Terraform templates describe a set of Azure resources and properties. For more information about how Terraform creates resources in Azure, see the [Azure Provider](#) topic on the Terraform site.

You may also find it helpful to review different deployment options in the [Reference Architecture for Pivotal Cloud Foundry on Azure](#).

Prerequisites

In addition to fulfilling the prerequisites listed in the [Installing Pivotal Cloud Foundry on Azure](#) topic, ensure you have the following:

- The [Terraform CLI](#)
- In your Azure project, ensure you have completed the steps in [Preparing to Deploy Ops Manager on Azure Using Terraform](#) to create a service principal.

Step 1: Download Templates and Edit Variables File

Before you can run Terraform commands to provision infrastructure resources, you must download the Azure Terraform Templates and create a Terraform template variables file as described below:

1. On [Pivotal Network](#), navigate to the Pivotal Application Service (PAS) release.
2. Download the Azure Terraform Templates ZIP file. You can use the templates to deploy PAS or.
3. Extract the contents of the ZIP file.
4. Move the extracted folder to the `workspace` directory on your local machine.
5. On the command line, navigate to the directory. For example:
   ```bash
   $ cd ~/workspace/pivotal-cf-terraforming-azure
   ```
6. Navigate to the `terraforming-pas` or `terraforming-pks` directory that contains the Terraform files for your runtime.
7. In the runtime directory, create a text file named `terraform.tfvars`.
8. Open the `terraform.tfvars` file and add the following:

   ```text
   subscription_id  = "YOUR-SUBSCRIPTION-ID"
   tenant_id        = "YOUR-TENANT-ID"
   client_id        = "YOUR-CLIENT-ID"
   client_secret    = "YOUR-CLIENT-SECRET"
   env_name         = "YOUR-ENVIRONMENT-NAME"
   env_short_name   = "YOUR-ENVIRONMENT-SHORTNAME"
   location         = "YOUR-AZURE-LOCATION"
   ops_manager_image_url = "YOUR-OPS-MAN-IMAGE-URI"
   dns_suffix       = "YOUR-DNS-SUFFIX"
   vm_admin_username = "YOUR-ADMIN-USERNAME"
   ```
9. Edit the values in the file according to the table below. For information about locating your Azure service principal values, see [Preparing to Deploy Ops Manager on Azure Using Terraform](#).

<table>
<thead>
<tr>
<th>Value to replace</th>
<th>Guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOUR-SUBSCRIPTION-ID</td>
<td>Enter the subscription ID of your Azure service principal. Terraform uses this ID when creating resources.</td>
</tr>
<tr>
<td>YOUR-TENANT-ID</td>
<td>Enter the tenant ID of your Azure service principal. Terraform uses this ID when creating resources.</td>
</tr>
</tbody>
</table>
YOUR-CLIENT-ID Enter the client ID of your Azure service principal. Terraform uses this ID when creating resources.

YOUR-CLIENT-SECRET Enter your Azure service client secret. Terraform requires this secret to create resources.

YOUR-ENVIRONMENT-NAME Enter a name to use to identify resources in Azure. Terraform prepends the names of the resources it creates with this environment name. This environment variable is also used to name the Azure resource group created for the deployment. Example: mypcf

YOUR-ENVIRONMENT-SHORTNAME Enter a name to use when creating storage accounts in Azure. Must be a-z only and no longer than 10 characters. Example: myazure

YOUR-AZURE-LOCATION Enter the name of the Azure location in which you want Terraform to create resources. Example: Central US

YOUR-OPS-MAN-IMAGE-URI Enter the URL for the Ops Manager Azure image you want to boot. You can find this code in the PDF included with the Ops Manager release on Pivotal Network.

YOUR-DNS-SUFFIX Enter a domain name to use as part of the system domain for your deployment. Terraform creates DNS records in Azure using YOUR-ENVIRONMENT-NAME and YOUR-DNS-SUFFIX. For example, if you enter example.com for your DNS suffix and have pcf as your environment name, Terraform creates DNS records at pcf.example.com.

YOUR-ADMIN-USERNAME Enter the admin username you want to use for your Ops Manager deployment.

---

**Step 2: (Optional) Add Variables for Isolation Segment**

Complete this section if you plan to deploy the Isolation Segment tile.

Note: You can see the configurable options by opening the variables.tf file and looking for variables with default values.

Add the following variable to your terraform.tfvars file. This causes Terraform to create an additional HTTP load balancer and DNS record to use for the Isolation Segment tile.

```
isolation_segment = "true"
```

**Step 3: Create Azure Resources with Terraform**

Follow these steps to use the Terraform CLI to create resources on Azure:

1. From the directory that contains the Terraform files, run `terraform init` to initialize the directory based on the information you specified in the terraform.tfvars file.

   $ terraform init

2. Run the following command to create the execution plan for Terraform.

   $ terraform plan -out=plan

3. Run the following command to execute the plan from the previous step. It may take several minutes for Terraform to create all the resources in Azure.

   $ terraform apply plan

**Step 4: Create DNS Record**

1. In a browser, navigate to the DNS provider for the DNS suffix you entered in your terraform.tfvars file.

2. Create a new name server (NS) record for your system domain. Your system domain is YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX.

3. In this record, enter the name servers included in env_dns_zone_name_servers from your Terraform output.
Next Steps

Proceed to the next step in the deployment, Configuring BOSH Director on Azure Using Terraform.
Configuring BOSH Director on Azure Using Terraform

This topic describes how to configure the BOSH Director tile within Ops Manager on Azure after Deploying Ops Manager to Azure.

Note: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Prerequisite

To complete the procedures in this topic, you must have access to the output generated when you ran `terraform apply` to create resources for this deployment. To view this output, run `terraform output` in the directory where you ran `terraform apply`. You use tidhe values in this output to configure the BOSH Director tile.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain name (FQDN) of the BOSH Director. Use the `ops_manager_dns` value from running `terraform output`. When redirected to the Internal Authentication page, you must complete the following steps:
   - Enter a Username, Password, and Password confirmation to create an Admin user.
   - Enter a Decryption passphrase and the Decryption passphrase confirmation. This passphrase encrypts the Ops Manager datastore, and is not
recoverable.

- If you are using an HTTP proxy or HTTPS proxy, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
- Read the End User License Agreement, and select the checkbox to accept the terms.
- Click Setup Authentication.

2. Log in to Ops Manager with the Admin username and password you created in the previous step.

SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.
3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **6a.** Ops Manager SAML service provider metadata: `https://OPS-MAN-FQDN:443/uaa/saml/metadata`
   - **6b.** BOSH Director SAML service provider metadata: `https://BOSH-IP-ADDRESS:8443/saml/metadata`

   **Note:** To retrieve your **BOSH-IP-ADDRESS**, navigate to the **Status** tab in the **BOSH Director** tile. Record the **BOSH Director** IP address.

7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on URL**: `https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN`
   - **Audience URI (SP Entity ID)**: `https://OPS-MAN-FQDN:443/uaa`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on URL**: `https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP`
   - **Audience URI (SP Entity ID)**: `https://BOSH-IP:8443`
   - **Name ID**: Email Address
   - SAML authentication requests are always signed

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

**LDAP Server**

- **For Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - `ldap://`: This specifies that the LDAP server uses an unencrypted connection.
  - `ldaps://`: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or
that you import a trusted certificate to the JVM truststore.

- **For LDAP Username and LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: cn=administrator,ou=Users,dc=example,dc=com

  **Note:** Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

**warning:** Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- **For User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: ou=Users,dc=example,dc=com

- **For User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter (cn=Smith) returns all objects with a common name equal to Smith.

In the LDAP search filter string that you use to configure your runtime, use {0} instead of the username. For example, use cn={0} to return all LDAP objects with the same common name as the username.

In addition to cn, other attributes commonly searched for and returned are mail, uid, and, in the case of Active Directory, sAMAccountName.

  **Note:** For instructions for testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry in the Pivotal Support Knowledge Base.

- **For Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: ou=Groups,dc=example,dc=com

- **For Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is member={0}.

- **For Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example mail.

- **For LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.

- From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
  - Automatically follow any referrals.
  - Ignore referrals and return partial result.
  - Throw exception for each referral and abort.

- **For Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

- Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

- If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

- Read the **End User License Agreement**, and select the checkbox to accept the terms.

- **Select Provision an admin client in the BOSH UAA**. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

- **Click Setup Authentication**.

- **Return to the BOSH Director tile**, and continue with the configuration steps below.
Step 2: Azure Config Page

1. Click the BOSH Director tile.

2. Select Azure Config.
3. Complete the following fields with information you obtained in the Preparing to Deploy Ops Manager on Azure Using Terraform topic.
4. Complete the following fields:

- Subscription ID: Enter the ID of your Azure subscription.
- Tenant ID: Enter your TENANT_ID.
- Application ID: Enter the APPLICATION_ID that you created in the Step 3: Create an AAD Application step of the Preparing to Deploy Ops Manager on Azure Using Terraform topic.
- Client Secret: Enter your CLIENT_SECRET.

5. For Cloud Storage Type, select Use Managed Disks. For Storage Account Type, select Premium_LRS which corresponds to SSD-based storage. See Azure Managed Disks Overview in the Microsoft documentation for more information.

6. For Default Security Group, enter the bosh_deployed_vms_security_group_name output from Terraform.

7. For SSH Public Key, enter the ops_manager_ssh_public_key output from Terraform.

8. For SSH Private Key, enter the ops_manager_ssh_private_key output from Terraform.

9. For Availability Mode, choose whether to use Availability Sets or Availability Zones. You can further configure this option in the Assign AZs and Networks Page.

   - For new installations of Ops Manager v2.5: Availability Zones are selected by default, but you can choose Availability Sets.
   - If you are upgrading from v2.4: You must use Availability Sets.

10. For Azure Environment, select Azure Commercial Cloud.

11. Click Save.

Step 3: Director Config Page

1. In Ops Manager, select Director Config.
2. In the **NTP Servers (comma delimited)** field, enter a comma-separated list of valid NTP servers.

   ![Diagram of Director Config](image)

   ```
   Note: The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify the value of this field.
   ```

3. Leave the **JMX Provider IP Address** field blank.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

   ```
   Note: Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank.
   ```

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the BOSH Resurrector functionality and increase your runtime availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   ```
   Note: If you intend to install, you must enable post-deploy scripts.
   ```

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. (Optional) Disable **Allow Legacy Agents** if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.
13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


   - **Service Key**: Enter your API service key from PagerDuty.
   - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

   - **Host**: Enter your email hostname.
   - **Port**: Enter your email port number.
   - **Domain**: Enter your domain.
   - **From**: Enter the address for the sender.
   - **Recipients**: Enter comma-separated addresses of intended recipients.
   - **Username**: Enter the username for your email server.
   - **Password**: Enter the password for your email server.
   - **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

   Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in **Preparing CredHub HSMs for Configuration**.

   **Note**: After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
### CredHub Encryption Provider

- **Internal**: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.
- **Luna HSM**: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Encryption Key Name</strong>:</td>
<td>Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.</td>
</tr>
<tr>
<td>2. <strong>Provider Partition</strong>:</td>
<td>The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in <a href="#">Preparing CredHub HSMs for Configuration</a>.</td>
</tr>
<tr>
<td>3. <strong>Provider Partition Password</strong>:</td>
<td></td>
</tr>
<tr>
<td>4. <strong>Provider Client Certificate</strong>:</td>
<td>The certificate that validates the identity of the HSM when CredHub connects as a client.</td>
</tr>
<tr>
<td>5. <strong>Provider Client Certificate Private Key</strong>:</td>
<td></td>
</tr>
<tr>
<td>6. <strong>HSM Host Address</strong>:</td>
<td></td>
</tr>
<tr>
<td>7. <strong>HSM Port Address</strong>:</td>
<td>If you do not know your port address, enter 1792.</td>
</tr>
<tr>
<td>8. <strong>Partition Serial Number</strong>:</td>
<td></td>
</tr>
<tr>
<td>9. <strong>HSM Certificate</strong>:</td>
<td>The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.</td>
</tr>
</tbody>
</table>

17. **Blobstore Location**: Select a Blobstore Location to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable...
and less secure, Pivotal recommends that you configure an external blobstore.

**Note:** After you deploy Ops Manager, you cannot change the blobstore location.

- **Internal**: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.
- **Enable TLS**: Select this checkbox to enable TLS to the blobstore.

  **Note:** If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

- **S3 Compatible Blobstore**: Select this option to use an external S3-compatible endpoint. Follow the procedures in [Sign up for Amazon S3](https://aws.amazon.com/s3/) and [Creating a Bucket](https://aws.amazon.com/s3/buckets/) in the AWS documentation. When you have created an S3 bucket, complete the following steps:

  1. **S3 Endpoint**: Navigate to the [Regions and Endpoints](https://aws.amazon.com/s3/#regions-and-endpoints) topic in the AWS documentation.

     a. Locate the endpoint for your region in the Amazon Simple Storage Service (S3) table and construct a URL using your region's endpoint. For example, if you are using the us-west-2 region, the URL you create would be `https://s3-us-west-2.amazonaws.com`. Enter this URL into the **S3 Endpoint** field.

     b. On a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the...
fully qualified domain name of Ops Manager.

c. Copy the custom public CA certificate you used to sign the S3 endpoint into `/etc/ssl/certs` on the Ops Manager VM.

d. On the Ops Manager VM, run `sudo update-ca-certificates -f -v` to import the custom CA certificate into the Ops Manager VM truststore.

![Note: You must also add this custom CA certificate into the Trusted Certificates field in the Security page. See Security Page for instructions.]

2. **Bucket Name**: Enter the name of the S3 bucket.
3. **Access Key** and **Secret Key**: Enter the keys you generated when creating your S3 bucket.
4. **Select V2 Signature or V4 Signature**: If you select V4 Signature, enter your **Region**.

![Note: AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see Authenticating Requests in the AWS documentation.]

- **GCS Blobstore**: Select this option to use an external GCS endpoint. To create a GCS bucket, you must have a GCS account. Follow the procedures in Creating Storage Buckets in the GCS documentation to create a GCS bucket. When you have created a GCS bucket, complete the following steps:

  1. **Bucket Name**: Enter the name of your GCS bucket.
  2. **Storage Class**: Select the storage class for your GCS bucket. See Storage Classes in the GCP documentation for more information.
  3. **Service Account Key**: Follow the steps in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually to download a JSON file with a private key. Enter the contents of the JSON file into the field.

18. For **Database Location**, select **Internal**.

19. (Optional) **Modify the Director Workers value**, which sets the number of workers available to execute Director tasks. This field defaults to 5.

20. (Optional) **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. (Optional) To add a custom URL for your BOSH Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director in the Pivotal Support Knowledge Base.

22. (Optional) **To set a custom banner** that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

23. (Optional) **Enter your comma-separated custom Identification Tags**. For example, `iaas:foundation1, hello:world`. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

![Note: Azure has limited space for identification tags. Pivotal recommends entering no more than five tags.]

24. **Click Save**.

**Step 4: Create Networks Page**

Select **Create Networks** and follow the procedures in this section to add the network configuration that you created for your VPC.
Create Network for PAS

Create the following three networks:

- PAS Infrastructure Network
- PAS Runtime Network
- PAS Services Network

Note: The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

PAS Infrastructure Network

Follow the steps below to create the infrastructure network.

1. (Optional) Select Enable ICMP checks to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

2. Click Add Network.

3. For Name, enter [infrastructure].

4. Under Subnets, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>[infrastructure]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Network Name</td>
<td>Enter NETWORK-NAME/SUBNET-NAME, where:</td>
</tr>
<tr>
<td></td>
<td>NETWORK-NAME is the network_name output from Terraform</td>
</tr>
<tr>
<td></td>
<td>SUBNET-NAME is the infrastructure_subnet_name output from Terraform</td>
</tr>
<tr>
<td>CIDR</td>
<td>Enter the CIDR listed under infrastructure_subnet_cidrs output from Terraform. For example, 10.0.8.0/26.</td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>Enter the first nine IP addresses of the subnet. For example, 10.0.8.1-10.0.8.9.</td>
</tr>
<tr>
<td>DNS</td>
<td>Enter [168.63.129.16].</td>
</tr>
<tr>
<td>Gateway</td>
<td>Enter the first IP address of the subnet. For example, 10.0.8.1.</td>
</tr>
</tbody>
</table>
5. Click **Save**.

Note: After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see Expanding Your Network with Additional Subnets.

### PAS Runtime Network

1. Click **Add Network**.

2. For **Name**, enter **pas**.

3. Under **Subnets**, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>pas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Network Name</td>
<td>Enter <code>[NETWORK-BASE/SUBNET-BASE]</code>, where:</td>
</tr>
<tr>
<td></td>
<td>- <code>[NETWORK-BASE]</code> is the <code>network_name</code> output from Terraform</td>
</tr>
<tr>
<td></td>
<td>- <code>[SUBNET-BASE]</code> is the <code>pas_subnet_name</code> output from Terraform</td>
</tr>
<tr>
<td>CIDR</td>
<td>Enter the CIDR listed under <code>pas_subnet_cidrs</code> output from Terraform. For example, <code>10.0.0.0/22</code></td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>Enter the first nine IP addresses of the subnet. For example, <code>10.0.0.1-10.0.0.9</code></td>
</tr>
</tbody>
</table>
DNS
Enter 168.63.129.16.

Gateway
Enter the first IP address of the subnet. For example, 10.0.0.1.

4. Click Save.

PAS Services Network
1. Click Add Network.
2. For Name, enter services.
3. Under Subnets, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Network Name</td>
<td>Enter NETWORK-NAME/SUBNET-NAME, where:</td>
</tr>
<tr>
<td></td>
<td>NETWORK-NAME is the network_name output from Terraform</td>
</tr>
<tr>
<td></td>
<td>SUBNET-NAME is the services_subnet_name output from Terraform</td>
</tr>
<tr>
<td>CIDR</td>
<td>Enter the CIDR listed under services_subnet_cidrs output from Terraform. For example, 10.0.4.0/22.</td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>Enter the first nine IP addresses of the subnet. For example, 10.0.4.1-10.0.4.9.</td>
</tr>
<tr>
<td>DNS</td>
<td>Enter 168.63.129.16.</td>
</tr>
<tr>
<td>Gateway</td>
<td>Enter the first IP address of the subnet. For example, 10.0.4.1.</td>
</tr>
</tbody>
</table>
4. Click **Save**. If **Enable ICMP checks** is not selected, you may see red warnings. You can ignore these warnings.

If you complete this step for PAS, proceed to **Assign AZs and Networks Page**.

---

### Create Network for

Create the following three networks:

- **Infrastructure Network**
- **Runtime Network**
- **Services Network**

**Note:** The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.

---

### Infrastructure Network

Follow the steps below to create the infrastructure network.

1. (Optional) Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

2. Click **Add Network**.

3. For **Name**, enter `infrastructure`.

4. Under **Subnets**, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Azure Network Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>infrastructure</strong></td>
<td>Enter <code>NETWORK-NAME/SUBNET-NAME</code>, where:</td>
</tr>
<tr>
<td></td>
<td>– <code>NETWORK-NAME</code> is the <code>network_name</code> output from Terraform</td>
</tr>
<tr>
<td></td>
<td>– <code>SUBNET-NAME</code> is the <code>infrastructure_subnet_name</code> output from Terraform</td>
</tr>
</tbody>
</table>
5. Click Save.

### Runtime Network
1. Click Add Network.
2. For Name, enter `pks`.
3. Under Subnets, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Azure Network Name</th>
<th>CIDR</th>
<th>Reserved IP Ranges</th>
<th>DNS</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>pks</td>
<td>Enter <code>NETWORK-NAME/SUBNET-NAME</code>, where:</td>
<td>Enter the CIDR listed under <code>pks_subnet_cidrs</code> output from Terraform. For example, <code>10.0.12.0/22</code>.</td>
<td>Enter the first nine IP addresses of the subnet. For example, <code>10.0.12.1-10.0.12.9</code>.</td>
<td>Enter <code>168.63.129.16</code>.</td>
<td>Enter the IP address listed under <code>pks_subnet_gateway</code> output from Terraform. This should be the first IP address of the subnet. For example, <code>10.0.12.1</code>.</td>
</tr>
</tbody>
</table>

4. Click Save.

### Services Network
1. Click Add Network.
2. For Name, enter `services`.
3. Under Subnets, create a subnet using the information in the following table:

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Azure Network Name</th>
<th>CIDR</th>
<th>Reserved IP Ranges</th>
<th>DNS</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>services</td>
<td>Enter <code>NETWORK-NAME/SUBNET-NAME</code>, where:</td>
<td>Enter the CIDR listed under <code>services_subnet_cidrs</code> output from Terraform. For example, <code>10.0.16.0/22</code>.</td>
<td>Enter the first nine IP addresses of the subnet. For example, <code>10.0.16.1-10.0.16.9</code>.</td>
<td>Enter <code>168.63.129.16</code>.</td>
<td>Enter the IP address listed under <code>services_subnet_gateway</code> output from Terraform. This should be the first IP address of the subnet. For example, <code>10.0.16.1</code>.</td>
</tr>
</tbody>
</table>

4. Click Save. If Enable ICMP checks is not selected, you may see red warnings. You can ignore these warnings.

### Step 5: Assign AZs and Networks Page
1. Select Assign AZs and Networks.

Note: The appearance of this page changes based on whether you are using Availability Zones or Availability Sets. For more information,
If you use Availability Zones, select one of the three available zones from the Singleton Availability Zone dropdown, then select a network from the Networks dropdown. The networks that appear in this list are networks that you created earlier in this procedure.

If you use Availability Sets, choose a network from the Networks dropdown. The networks that appear in this list are networks you created earlier in this procedure.

2. Click Save.

Step 6: Security Page


2. In Trusted Certificates, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEF12345678ABCDFF12345678ABCDFF12345678ABCDFF12345678ABCDEF
GH12345678ABCDFF12345678ABCDFF12345678ABCDEF12345678...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
BCDEFG12345678ABCDFF12345678ABCDFF12345678ABCDFF12345678ABCDEF
GH12345678ABCDFF12345678ABCDFF12345678ABCDEF12345678...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
CDEFG12345678ABCDFF12345678ABCDFF12345678ABCDFF12345678ABCDEF
GH12345678ABCDFF12345678ABCDFF12345678ABCDEF12345678...
-----END CERTIFICATE-----
```

2. In Trusted Certificates, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEF12345678ABCDFF12345678ABCDFF12345678ABCDFF12345678ABCDEF
GH12345678ABCDFF12345678ABCDFF12345678ABCDEF12345678...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
BCDEFG12345678ABCDFF12345678ABCDFF12345678ABCDFF12345678ABCDEF
GH12345678ABCDFF12345678ABCDFF12345678ABCDEF12345678...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
CDEFG12345678ABCDFF12345678ABCDFF12345678ABCDFF12345678ABCDEF
GH12345678ABCDFF12345678ABCDFF12345678ABCDEF12345678...
-----END CERTIFICATE-----
```

Note: If you want to use Docker registries to run PAS app instances in Docker containers, enter the certificate for your private Docker registry in this field. See Using Docker Registries for more information about running app instances in PAS using Docker registries.

3. (Optional) Select the Include OpsManager Root CA in Trusted Certs checkbox to include the Ops Manager root CA in the Trusted Certificates field.
BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose Generate passwords or Use default BOSH password. Pivotal recommends that you use the Generate passwords option for greater security.

5. Click Save. To view your saved Director password, click the Credentials tab.

Step 7: BOSH DNS Config Page

1. Select BOSH DNS Config.

2. (Optional) In Excluded Recursors, enter a list of prohibited recursor addresses.

3. (Optional) In Recursor Timeout, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   Note: This time limit must include one of the Go parse duration time units. For example, entering \(5s\) sets the timeout limit to five seconds. For more information about supported time units, see `func ParseDuration` in the Go Programming Language documentation.

4. (Optional) In Handlers, enter a list of custom domain handlers in JSON format.

5. Click Save.

Step 8: Syslog Page
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.
7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** field. This field requires the rainscript syntax.

10. Click **Save Syslog Settings**.

### Step 9: Resource Config Page

1. Select **Resource Config**.

2. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the dropdown to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

   **Note:** Pivotal recommends provisioning a BOSH Director VM with at least 8 GB memory.

   **Note:** If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the new recommended allocation.

   **Note:** If you install PAS for Windows, provision your **Master Compilation Job** with at least 100 GB of disk space.

3. (Optional) For **Load Balancers**, enter your Azure application gateway name for each associated job. To create an application gateway, follow the procedures in Create an application gateway with SSL termination using Azure PowerShell from the Azure documentation. When you create the application gateway, associate the gateway’s IP with your system domain.

4. Click **Save**.

### Step 10: (Optional) Add Custom VM Extensions

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see Managing Custom VM Extensions.

### Step 11: Complete the BOSH Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Review Pending Changes**, then **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.
3. BOSH Director installs. This may take a few moments. When the installation process successfully completes, the Changes Applied window appears.

Next Steps

After you complete this procedure, do one of the following:

- If you are deploying PAS, follow the procedures in Configuring PAS.
- If you are deploying , follow the procedures in Creating Managed Identities in Azure for [1].
Troubleshooting PCF on Azure

Installation Issues

Cannot Copy the Ops Manager Image

Symptom

Cannot copy the Ops Manager image into your storage account as part of Step 4: Boot Ops Manager in Deploying Ops Manager on Azure Manually.

Explanation

You have an outdated version of the Azure CLI. You need the Azure CLI version 2.0.0 or greater. Run `az --version` from the command line to display your current Azure CLI version.

Solution

Install the Azure CLI 2.0 by following the instructions for your operating system in the Azure documentation.

Deployment Fails at “create-env”

Symptom

After clicking Review Pending Changes, then Apply Changes to install Ops Manager and PAS, the deployment fails at `create-env` with an error message similar to the following:

```
Command 'deploy' failed:
Deploying:
Creating instance 'bosh/0':
Waiting until instance is ready:
  Starting SSH tunnel:
    Parsing private key file '/tmp/bosh_ec2_private_key.pem':
   asn1: structure error: tags don't match (16 vs {class:3 tag:28 length:127}
isCompound: false) {optional: false explicit: false application: false
    defaultValue: <nil> tag:<nil> stringType: 0 timeType: 0 set: false omitEmpty: false}
  pkcs1PrivateKey @2
=====
2016-09-29 16:28:22 UTC Finished "bosh create-env"
/var/tempest/workspaces/default/deployments/bosh.yml";
Duration: 328s; Exit Status: 1
Exited with 1.
```

Explanation

You provided a passphrase when creating your key pair in Step 4: Boot Ops Manager of Deploying Ops Manager on Azure Manually.

Solution

Create a new key pair with no passphrase and redo the installation. See Step 4: Boot Ops Manager in Deploying Ops Manager on Azure Manually.
Insufficient External Database Permissions

Upgrade issues can be caused when the external database user used for the network policy DB is given insufficient permissions. To avoid this upgrade issue, ensure that the networkpolicyserver database user has the `ALL PRIVILEGES` permission.

Operation Issues

Slow Performance or Timeouts

Symptom

Developers suffer from slow performance or timeouts when pushing or managing apps, and end users suffer from slow performance or timeouts when accessing apps.

Explanation

The Azure Load Balancer (ALB) disconnects active TCP connections lying idle for over four minutes.

Solution

To mitigate slow performance or timeouts, the default value of the `Router Timeout to Backends (in seconds)` field is set to 900 seconds. This default value is set high to mitigate performance issues but operators should tune this parameter to fit their infrastructure.

To edit the `Router Timeout to Backends (in seconds)` field:

1. Select the Pivotal Application Service (PAS) tile that is located within your Installation Dashboard.
2. Select the Networking tab.
3. Enter your desired time, in seconds, within the `Router Timeout to Backends (in seconds)` field.
4. Click Save.

Service Instance Creation Times Out

Symptom

You are unable to provision a service instance of a Java or Go service. `cf create-service` fails with error `Failure provisioning service instance... Timed out after 8 minutes...` or similar.

Explanation

HTTP libraries for Java and Go, running with default settings, prune idle (for 240 seconds) connections from their connection pool without sending a TCP reset message back to the client service broker. This removes the ability of the broker to provision new service instances.
Solution

In the PAS tile Configure Networking pane, set Frontend Idle Timeout to 240 seconds or less, so that Cloud Foundry regenerates the front-end connection before it times out.

See the Knowledge Base article Azure Networking Connection Idle for more than Four minutes for details.
Deleting a PCF on Azure Installation

Page last updated:

When you deploy Pivotal Cloud Foundry (PCF) to Azure, you provision a set of resources. This topic describes how to delete the resources associated with a PCF deployment.

The fastest way to remove resources is to delete the resource group, or resource groups, associated with your PCF on Azure installation.

Delete the Resource Group

Perform the following steps to delete a resource group:

1. Navigate to the Azure Portal.

2. Within your subscription, select Resource Groups.

3. Click on the resource group you wish to delete.

4. In the details pane for the resource group, click on the trash can icon. Review the information in the confirmation screen before proceeding.

5. To confirm deletion, type in the resource group name and click Delete.

For more information about managing resource groups in Azure, see the Azure documentation.
Upgrading BOSH Director on Azure

Page last updated:

This topic describes how to upgrade BOSH Director for Pivotal Cloud Foundry (PCF) on Azure.

Follow the procedures below as part of the upgrade process documented in the [Upgrading Pivotal Cloud Foundry](#) topic.

**Step 1: Export Environment Variables**

1. Install the Azure CLI v2.0 by following the instructions for your operating system in the [Azure documentation](#).
2. Set your cloud:

   ```
   $ az cloud set --name AzureCloud
   ```

   If you deployed PCF in an environment other than Azure Cloud, consult the following list:

   - For Azure China, replace `AzureCloud` with `AzureChinaCloud`. If logging in to `AzureChinaCloud` fails with a `CERT_UNTRUSTED` error, use the latest version of node, 4.x or later.
   - For Azure Government Cloud, replace `AzureCloud` with `AzureUSGovernment`.
   - For Azure Germany, replace `AzureCloud` with `AzureGermanCloud`.
3. Log in:

   ```
   $ az login
   ```

   Authenticate by navigating to the URL in the output, entering the provided code, and clicking your account.
4. Ensure that the following environment variables are set to the names of the resources you created when deploying Ops Manager as part of the procedures in [Deploying Ops Manager on Azure Manually](#).

   - `$RESOURCE_GROUP`: This should be set to the name of your resource group. Run `az group list` to list the resource groups for your subscription.
   - `$LOCATION`: This should be set to your location, such as `westus`. For a list of available locations, run `az account list-locations`.
   - `$STORAGE_NAME`: This should be set to your BOSH storage account name. Run `az storage account list` to list your storage accounts.
5. Retrieve the connection string for the account.

   ```
   $ az storage account show-connection-string \
   --name $STORAGE_NAME --resource-group $RESOURCE_GROUP
   ```

   The command returns output similar to the following:

   ```
   
   
   ```

   6. From the `data` field in the output above, record the full value of `connectionString` from the output above, starting with and including `DefaultEndpointsProtocol=`.

   7. Export the value of `connectionString` as the environment variable `$AZURE_STORAGE_CONNECTION_STRING`.

      ```
      $ export AZURE_STORAGE_CONNECTION_STRING="YOUR-ACCOUNT-KEY-STRING"
      ```

**Step 2: Set Up Ops Manager**

1. Navigate to [Pivotal Network](#) and download the release of Pivotal Cloud Foundry Ops Manager for Azure you want to upgrade to.
2. View the downloaded PDF and locate the Ops Manager image URL appropriate for your region.

3. Export the Ops Manager image URL as an environment variable.

   ```bash
   export OPS_MAN_IMAGE_URL="YOUR-OPS-MAN-IMAGE-URL"
   ```

   This command overrides the old Ops Manager image URL with the new Ops Manager image URL.

4. Copy the Ops Manager image into your storage account. For compatibility when upgrading to future versions of Ops Manager, choose a unique name for the image that includes the Ops Manager version number. For example, replace `opsman-image-version` in the following examples with `opsman-image-2.0.1`.

   ```bash
   $ az storage blob copy start --source-uri $OPS_MAN_IMAGE_URL
   --connection-string $AZURE_STORAGE_CONNECTION_STRING
   --destination-container opsmanager
   --destination-blob opsman-image-version.vhd
   ```

   When `status` reads `success`, continue to the next step.

5. Copying the image may take several minutes. Run the following command and examine the output under "copy":

   ```bash
   $ az storage blob show --name opsman-image-version.vhd
   --container-name opsmanager
   --account-name $STORAGE_NAME
   ...
   "copy": {
       "completionTime": "2017-06-26T22:24:11+00:00",
       "id": "b9c8b272-a562-4574-baa6-f1a04afcefdf",
       "progress": "53687091712/53687091712",
       "source": "https://opsmanagerwestus.blob.core.windows.net/images/ops-manager-2.0.x.vhd",
       "status": "success",
       "statusDescription": null
   },
   ```

   Step 3: Configure IP Address

   You have two choices for the Ops Manager IP address. Choose one of the following:

   - **Reuse the existing dynamic public IP address.**
   - **Use a new dynamic public IP address.**

   **Reuse Existing Dynamic Public IP Address**

   1. List your VMs and record the name of your Ops Manager VM:

   ```bash
   $ az vm list
   ```

   2. Delete your old Ops Manager VM:

   ```bash
   $ az vm delete --name YOUR-OPS-MAN-VM --resource-group $RESOURCE_GROUP
   ```

   3. List your network interfaces and record the name of the Ops Manager network interface:

   ```bash
   $ az network interface list --resource-group $RESOURCE_GROUP
   ```

   **Use a New Dynamic Public IP Address**

   1. Create a new public IP address named `ops-manager-ip-new`
2. Record the value for `ipAddress` from the output above. This is the public IP address of Ops Manager.

3. Create a network interface for Ops Manager.

   ```
   $ az network nic create --vnet-name pcf-net
   --subnet pcf --network-security-group opsmgr-nsg
   --private-ip-address 10.0.0.5
   --public-ip-address ops-manager-ip-new
   --resource-group $RESOURCE_GROUP
   --name opsmgr-nic-new --location $LOCATION
   ```

4. Shut down your old Ops Manager VM, if it still exists:

   ```
   $ az vm deallocate --name ops-manager --resource-group $RESOURCE_GROUP
   ```

   If your Ops Manager VM is not named `ops-manager`, provide the correct name. To list all VMs in your account, use `az vm list`.

5. Update your DNS record to point to your new public IP address of Ops Manager.

---

**Step 4: Boot Ops Manager**

1. If you want to use the key pair from your previous Ops Manager, locate the path to the file on your local machine. If you want to create a new key pair, enter the following command:

   ```
   $ ssh-keygen -t rsa -f opsman-
   ```

   When prompted for a passphrase, press the **enter** key to provide an empty passphrase.

2. Create the Ops Manager VM.

   - If you are using unmanaged disks, run the following command to create your Ops Manager VM, replacing `PATH-TO-PUBLIC-KEY` with the path to your public key .pub file, and replacing `opsman-image-version` with the version of Ops Manager you are deploying, for example `opsman-image-2.0.1`:

   ```
   $ az vm create --name opsman-version --resource-group $RESOURCE_GROUP
   --location SLOCATION
   --nics opsmgr-nic
   --os-disk-name opsmgr-osdisk
   --os-disk-size-gb 128
   --os-type Linux
   --use-unmanaged-disk
   --storage-account $STORAGE_NAME
   --storage-container-name opsmanager
   --admin-username ubuntu
   --ssh-key-value PATH-TO-PUBLIC-KEY
   ```
Replace [my-azure-instance.com] with the URL of your Azure instance. Find the complete source URL in the Azure UI by viewing the Blob properties of the Ops Manager image you created earlier in this procedure.

- If you are using Azure managed disks, perform the following steps, replacing [opsman-image-version] with the version of Ops Manager you are deploying, for example [opsman-image-2.0.1]:

  1. Create a managed image from the Ops Manager VHD file:

     ```
     $ az image create --resource-group $RESOURCE_GROUP
     --name opsman-image-version
     --source https://$STORAGE_NAME.blob.core.windows.net/opsmanager/opsman-image-version.vhd
     --location $LOCATION
     --os-type Linux
     ```

     If you are using Azure China, Azure Government Cloud, or Azure Germany, replace blob.core.windows.net with the following:

     - For Azure China, use blob.core.chinacloudapi.cn. See the Azure documentation for more information.
     - For Azure Government Cloud, use blob.core.usgovcloudapi.net. See the Azure documentation for more information.
     - For Azure Germany, use blob.core.cloudapi.de. See the Azure documentation for more information.

  2. Create your Ops Manager VM, replacing [PATH-TO-PUBLIC-KEY] with the path to your public key .pub file, and replacing [opsman-version] and [opsman-image-version] with the version of Ops Manager you are deploying, for example [opsman-2.0.1], [opsman-image-2.0.1-osdisk], and [opsman-image-2.0.1]:

     ```
     $ az vm create --name opsman-version
     --resource-group $RESOURCE_GROUP
     --location $LOCATION
     --nics opsman-nic
     --image opsman-image-version
     --os-disk-name opsman-version-osdisk
     --admin-username ubuntu
     --size Standard_DS2_v2
     --storage-sku Standard_LRS
     --ssh-key-value PATH-TO-PUBLIC-KEY
     ```

  3. If you have deployed more than one tile in this Ops Manager installation, perform the following steps to increase the size of the Ops Manager VM disk. You can repeat this process and increase the disk again at a later time if necessary.

     **Note:** If you use Azure Stack, you must increase the Ops Manager VM disk size using the Azure Stack UI.

     a. Run the following command to stop the VM and detach the disk, replacing [opsman-version] with the version of Ops Manager you are deploying, for example [opsman-2.0.1]:

        ```
        $ az vm deallocate --name opsman-version
        --resource-group $RESOURCE_GROUP
        ```

     b. Run the following command to resize the disk to 128 GB, replacing [opsman-version] with the version of Ops Manager you are deploying, for example [opsman-2.0.1]:

        ```
        $ az disk update --size-gb 128 --name opsman-version-osdisk
        --resource-group $RESOURCE_GROUP
        ```

     c. Run the following command to start the VM, replacing [opsman-version] with the version of Ops Manager you are deploying, for example [opsman-2.0.1]:

        ```
        $ az vm start --name opsman-version
        --resource-group $RESOURCE_GROUP
        ```
Installing Pivotal Cloud Foundry on GCP

This topic describes how to install Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP). It includes resource requirements, prerequisites, instructions for installing PCF on GCP, and additional resources.

Overview

You can install PCF on GCP with either the Pivotal Application Service (PAS) or Pivotal Container Service (PKS) runtime. There are resource requirements specific to each runtime. Ensure you meet the requirements for your runtime and the requirements specific to GCP before installing PCF on GCP.

Requirements

This section lists the following resource requirements for installing PCF on GCP:

- General PCF resource requirements. See PCF Resource Requirements.
- GCP-specific resource requirements. See GCP Resource Requirements.

PCF Resource Requirements

This section lists PCF resource requirements for installing PCF on GCP. It includes general PCF resource requirements for both the PAS and PKS runtimes.

View one of the following, depending on your PCF runtime:

- PAS-specific PCF resource requirements. See PAS Resource Requirements.
- PKS-specific PCF resource requirements. See PKS Resource Requirements.

PAS Resource Requirements

The following are general resource requirements for deploying and managing a PCF deployment with Ops Manager and PAS:

- PAS requires sufficient IP allocation. The following lists the minimum required IP allocations:
  - One static IP address for either HAProxy or one of your Gorouters
  - One static IP address for each job in the Ops Manager tile. See the Ops Manager Resource Config pane for each tile for a full list.
  - One static IP address for each job listed below:
    - Consul
    - NATS
    - File Storage
    - MySQL Proxy
    - MySQL Server
    - Backup Restore Node
    - HAProxy
    - Router
    - MySQL Monitor
    - Diego Brain
    - TCP Router
  - One IP for each VM instance created by the service.
  - An additional IP address for each compilation worker. Use the following formula to determine the total IPs required:
    \[ \text{IPs needed} = \text{static IPs} + \text{VM instances} + \text{compilation workers} \]
  - Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

- Pivotal recommends using a network without DHCP for deploying PAS VMs.
PKS Resource Requirements

For PKS-specific resource requirements, see GCP Prerequisites and Resource Requirements.

GCP Resource Requirements

The following are GCP-specific resource requirements for installing PCF on GCP:

- Installing PCF on GCP requires a minimum of the following VM instance limits in your GCP account. The number of VMs required depends on the number of tiles and availability zones you plan to deploy. The following VM guidelines apply to the PAS and Small Footprint PAS runtimes:

  - **PAS**: At a minimum, a new GCP deployment requires the following VMs for PAS and Ops Manager:

    | VM Count | vCPU Count per VM | RAM (GB) |
    |----------|-------------------|----------|
    | 30       | 1                 | 1        |
    | 3        | 1                 | 2        |
    | 4        | 2                 | 4        |
    | 3        | 2                 | 8        |
    | 3        | 4                 | 16       |

  - **Small Footprint PAS**: At a minimum, a new GCP deployment requires the following VMs to run Small Footprint PAS:

    | VM Type     | VM Count | vCPU Count per VM | RAM (GB) | Notes |
    |-------------|----------|-------------------|----------|-------|
    | Small Footprint PAS | micro | 12 | 1 | 1 | Add 1 to VM count if using HAProx | |
    |              | small   | 3   | 1 | 2 |                                  | |
    |              | xlarge.disk | 1   | 4 | 16 |                                  | |
    |              | xlarge   | 1   | 4 | 16 |                                  | |
    |              | medium.mem | 1   | 1 | 6  |                                  | |
    |              | large.disk | 1   | 2 | 8  |                                  | |
    |              | large.disk | 1   | 2 | 8  |                                  | |
    |              | large.cpu | 4   | 4 | 4  |                                  | |

  - **Ops Manager**: For more information, see Scaling PAS.

- **PKS**: See GCP Prerequisites and Resource Requirements.

(PAS-only) Your GCP project must have sufficient quota to deploy all the VMs needed to install PCF with PAS. For a list of suggested quotas, see Recommended GCP Quotas.

Prerequisites

To install PCF on GCP, you must do the following:

- Install the Google Cloud SDK on your machine and authenticate it to your GCP account. To download the Google Cloud SDK, see Google Cloud SDK.

- Increase or remove the VM instance limits in your GCP account. For VM instance requirements, see GCP Resource Requirements.

- Update your GCP account with the following required permissions:

  - Compute Engine > Compute Instances Admin (beta)
  - Compute Engine > Compute Network Admin
  - Compute Engine > Compute Security Admin
To use Google Cloud Storage (GCS) for Cloud Controller file storage, the following permission is required to create buckets:

- Storage > Storage Admin

To use Cloud DNS, the following permission is required to add and modify DNS entries:

- Project > Editor

Create an SSL certificate for your PCF domain.

**Note:** To deploy PCF to a production environment, you must obtain a certificate from a certificate authority. Pivotal recommends using a self-signed certificate generated by Ops Manager for development and testing purposes only.

Assign administrative rights to a domain for PCF. You need to be able to add wildcard records to this domain.

Create a wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`. Then, create at least one wildcard TLS certificate that matches the DNS record you configured.

**Note:** With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer. For example, if you create a DNS record `*.example.com` pointing to your router, every app deployed to the `example.com` domain resolves to the IP address of your router.

Create one or more NTP servers, if the NTP servers are not already provided by your GCP project.

Install the most recent version of one of the following CLIs, depending on your PCF runtime:

- PKS: The PKS CLI. See [Installing the PKS CLI](https://github.com).

(PAS-only) Request a quota increase for your GCP project. For GCP quota requirements, see [GCP Resource Requirements](https://github.com). To request an increase, see [Quotas](https://github.com) in the GCP console.

(PAS-only) Configure sufficient IP allocation. For more information about IP allocation requirements, see [PAS Resource Requirements](https://github.com).

(Optional) (PAS-only) Configure external storage. Pivotal recommends using external storage if possible. For more information about how file storage location affects platform performance and stability during upgrades, see [Configure File Storage](https://github.com).

(Optional) (PAS and Ops Manager-only) Configure external databases. Pivotal recommends using external databases in production deployments for BOSH Director and PAS. An external database must be configured to use the UTC timezone.

(Optional) (PAS and Ops Manager-only) Configure external user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.

**Install PCF on GCP**

You can install PCF on GCP either manually or using Terraform.

To install PCF on GCP, do one of the following:

- Install PCF on GCP manually. See [Installing PCF on GCP Manually](https://github.com).
- Install PCF on GCP using Terraform. See [Install PCF on GCP Using Terraform](https://github.com).

**Additional Resources**

The following are additional resources related to installing PCF on GCP:

- For general authentication guidelines for GCP, see [GCP authentication documentation](https://github.com).
- For more information about troubleshooting the PCF on GCP infrastructure, see [Troubleshooting PCF on GCP](https://github.com).
- For production-level deployment options for PCF on GCP, see [Reference Architecture for Pivotal Cloud Foundry on GCP](https://github.com).
- For general certificate requirements for deploying PCF, see [Certificate Requirements](https://github.com).
Recommended GCP Quotas

Default quotas on a new GCP subscription do not have enough quota for a typical production-level PCF deployment.

The following table lists recommended resource quotas for a single PCF deployment.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Resource</th>
<th>Suggested Minimum Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs</td>
<td>Regional</td>
<td>150*</td>
</tr>
<tr>
<td>Firewall rules</td>
<td>Global</td>
<td>15</td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>Global</td>
<td>15</td>
</tr>
<tr>
<td>Backend services</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Health checks</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Images</td>
<td>Global</td>
<td>10</td>
</tr>
<tr>
<td>Static IP addresses**</td>
<td>Regional</td>
<td>5</td>
</tr>
<tr>
<td>In-use IP addresses</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>In-use IP addresses**</td>
<td>Regional</td>
<td>5</td>
</tr>
<tr>
<td>Networks</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Subnetworks</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Routes</td>
<td>Global</td>
<td>20</td>
</tr>
<tr>
<td>Target pools</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Target HTTP proxies</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Target HTTPS proxies</td>
<td>Global</td>
<td>5</td>
</tr>
<tr>
<td>Persistent Disk Standard (GB)</td>
<td>Regional</td>
<td>15,000</td>
</tr>
</tbody>
</table>

* Assuming a deployment with 100 app instances.
** Assuming a SNAT topology.

To view production-level deployment options for PCF on GCP, see the [Reference Architecture for Pivotal Cloud Foundry on GCP](https://www.pivotal.io/products/cloud-foundry-reference-architecture-gcp/).

For instructions on how to set up GCP resources required to deploy PCF, see [Preparing to Deploy Ops Manager on GCP Manually](https://www.pivotal.io/products/cloud-foundry-preparing-gcp-manually/).
Installing PCF on GCP Manually
This topic explains how to install PCF on GCP manually.

Overview
You can install PCF on GCP with either the Pivotal Application Service (PAS) or Pivotal Container Service (PKS) runtime.

To install PCF on GCP manually, do one of the following:

- Install with PAS. See Install Manually with PAS.
- Install with PKS. See Install Manually with PKS.

Install Manually with PAS
To install manually with PAS, do the following:

1. Prepare to deploy Ops Manager. See Preparing to Deploy Ops Manager on GCP Manually.
2. Deploy Ops Manager. See Deploying Ops Manager on GCP Manually.
3. Configure BOSH Director. See Configuring BOSH Director on GCP Manually.
4. (Optional). Configure a shared VPC. See (Optional) Configuring a Shared VPC on GCP.
5. Configure PAS. See Configuring PAS.

Install Manually with PKS
To install manually with PKS, do the following:

1. Prepare to deploy Ops Manager. See Preparing to Deploy Ops Manager on GCP Manually.
2. Deploy Ops Manager. See Deploying Ops Manager on GCP Manually.
3. Configure BOSH Director. See Configuring BOSH Director on GCP Manually.
4. (Optional). Configure a shared VPC. See (Optional) Configuring a Shared VPC on GCP.
5. Configure PKS. See Installing PKS on GCP.
Preparing to Deploy Ops Manager on GCP Manually

This topic describes the preparation steps required to install Ops Manager for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

Prerequisites

Before you prepare your Ops Manager installation, do the following depending on the runtime you intend to deploy:

- If you are deploying Pivotal Application Service (PAS), see PCF on GCP Requirements.
- If you are deploying , see GCP Prerequisites and Resource Requirements.

Configuration and Components

This section outlines high-level infrastructure options for PCF on GCP. A PCF deployment includes Ops Manager and your chosen runtime. For example, both Ops Manager with PAS and Ops Manager with are PCF deployments. For more information, review the deployment options and recommendations in Reference Architecture for Pivotal Cloud Foundry on GCP.

You can deploy PCF using one of two main configurations on a GCP virtual private cloud (VPC):

- A single-project configuration that gives Ops Manager full access to VPC resources
- A shared VPC configuration in which Ops Manager shares VPC resources

See Shared vs Single-Project VPCs in the Reference Architecture for Pivotal Cloud Foundry on GCP topic for a full discussion and recommendations.

When deploying PCF on GCP, Pivotal recommends using the following GCP components:

- Google Cloud SQL for external database services
- NAT Gateway Instances to limit the number of VMs with public IP addresses
- Google Cloud Storage for external file storage

Step 1: Set up IAM Service Accounts

Ops Manager uses IAM service accounts to access GCP resources.

For a single-project installation: Complete the following steps to create a service account for Ops Manager.

For a shared-VPC installation: Complete the following steps twice, to create a host account and service account for Ops Manager.

1. From the GCP console, select IAM & Admin, then Service accounts.

2. Click Create Service Account:
   - Service account name: Enter a name. For example, _bosh_.
   - Role: Select the following roles:
     - Service Accounts > Service Account User
     - Service Accounts > Service Account Token Creator
     - Compute Engine > Compute Instance Admin (v1)
     - Compute Engine > Compute Network Admin
     - Compute Engine > Compute Storage Admin
     - Storage > Storage Admin

   Note: You must scroll down in the pop-up windows to select all required roles.

   The Service Account User role is only required if you plan to use The Ops Manager VM Service Account to deploy Ops Manager.

   For more information about The Ops Manager VM Service Account, see Step 2: Google Cloud Platform Config in Configuring BOSH Director on GCP.
Service account ID: The field automatically generates a unique ID based on the username.
- Furnish a new private key: Select this checkbox and JSON as the Key type.

3. Click Create. Your browser automatically downloads a JSON file with a private key for this account. Save this file in a secure location.

Step 2: Enable Google Cloud APIs

Ops Manager manages GCP resources using the Google Compute Engine and Cloud Resource Manager APIs. To enable these APIs, do the following:


2. In the console, navigate to the GCP projects where you want to install Ops Manager.
   - For a single-project installation, complete the following steps for the Ops Manager project.
   - For a shared-VPC installation, complete the following steps for both host and service projects, to enable them to access the Google Cloud API.

3. Select API Manager > Library.

4. Under Google Cloud APIs, select Compute Engine API.

5. On the Google Compute Engine API page, click Enable.

6. In the search field, enter Google Cloud Resource Manager API.


8. To verify that the APIs have been enabled, perform the following steps:
Step 3: Create a GCP Network with Subnets

1. Log in to the GCP console.

2. Navigate to the GCP project where you want to install Ops Manager. For a shared-VPC installation, navigate to the host project.

3. Select VPC network, then CREATE VPC NETWORK.

4. In the Name field, enter a name of your choice for the VPC network. This name helps you identify resources for this deployment in the GCP console. Network names must be lowercase. For example, `pcf-virt-net`.

   a. Under Subnets, complete the form as follows to create an infrastructure subnet for Ops Manager and NAT instances:

      | Name                     | Example                                      |
      |--------------------------|----------------------------------------------|
      | `pcf-infrastructure-subnet-GCP-REGION` | `pcf-infrastructure-subnet-us-west1`          |
      | Region                   | A region that supports three availability zones. For help selecting the correct region for your deployment, see the Google documentation about regions and zones. |
      | IP address range         | A CIDR ending in /26                          |
      |                          | Example: 192.168.101.0/26                    |

See the following image for an example:
b. Click Add subnet to add a second subnet for the BOSH Director and components specific to your runtime. Complete the form as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>pcf-subnet-infrastructure-us-west1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>us-west1</td>
</tr>
<tr>
<td>IP address range</td>
<td>192.168.101.0/26</td>
</tr>
</tbody>
</table>

Create secondary IP range

Private Google access

- On
- Off

Flow logs

- On
- Off

Done Cancel

Note: For deployments that do not use external IP addresses, enable Private Google access to allow your runtime to make API calls to Google services.

c. Click Add subnet to add a third Subnet with the following details:

<table>
<thead>
<tr>
<th>Name</th>
<th>pcf-services-subnet-GCP-REGION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>The same region you selected for the infrastructure subnet</td>
</tr>
<tr>
<td>IP address range</td>
<td>A CIDR ending in /22</td>
</tr>
<tr>
<td></td>
<td>Example: 192.168.16.0/22</td>
</tr>
</tbody>
</table>
IP address range

A CIDR in /22

Example: 192.168.20.0/22

See the following image for an example:

<table>
<thead>
<tr>
<th>VPC networks</th>
<th>CREATE VPC NETWORK</th>
<th>REFRESH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Region</td>
<td>Subnets</td>
</tr>
<tr>
<td>pcf-vnet-network</td>
<td>us-west1</td>
<td>pcf-subnet-us-west1</td>
</tr>
<tr>
<td></td>
<td>us-west1</td>
<td>pcf-subnet-infrastructure-us-west1</td>
</tr>
<tr>
<td></td>
<td>us-west1</td>
<td>pcf-subnet-services-us-west1</td>
</tr>
</tbody>
</table>

5. Under Dynamic routing mode, leave Regional selected.

6. Click Create.

Step 4: Create NAT Instances

Use NAT instances when you want to expose only a minimal number of public IP addresses.

Creating NAT instances permits internet access from cluster VMs. You might, for example, need this internet access for pulling Docker images or enabling internet access for your workloads.

For more information, see Reference Architecture for Pivotal Cloud Foundry on GCP and GCP documentation.

1. In the GCP console, with your single project or shared-VPC host project selected, navigate to Compute Engine > VM instances.

   ![Compute Engine](image)

2. Click CREATE INSTANCE

3. Complete the following fields:

   - **Name**: Enter `pcf-nat-gateway-pri`
     This is the first, or primary, of three NAT instances you need. If you use a single AZ, you need only one NAT instance.
   - **Zone**: Select the first zone from your region.
     Example: For region `us-west1`, select zone `us-west1-a`.
   - **Machine type**: Select `n1-standard-4`.
   - **Boot disk**: Click Change and select `Ubuntu 14.04 LTS`.
4. Expand the additional configuration fields by clicking Management, disks, networking, SSH keys.

![Screenshot of instance creation page]

a. In the Startup script field under Automation, enter the following text:

```bash
#!/bin/bash
sudo sysctl -w net.ipv4.ip_forward=1
sudo sh -c 'echo net.ipv4.ip_forward=1 >> /etc/sysctl.conf'
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

5. Click Networking to open additional network configuration fields:

a. In the Network tags field, add the following: nat-traverse and pcf-nat-instance.

![Screenshot of network tags]

b. Click on the Networking tab and the pencil icon to edit the Network interface.

c. For Network, select pcf-virt-net. You created this network in Step 1: Create a GCP Network with Subnets.

d. For Subnetwork, select pcf-infrastructure-subnet-GCP-REGION.

e. For Primary internal IP, select Ephemeral (Custom). Enter an IP address, for example, 192.168.101.2, in the Custom ephemeral IP address field. The IP address must meet the following requirements:

- The IP address must exist in the CIDR range you set for the pcf-infrastructure-subnet-GCP-REGION subnet.
- The IP address must exist in a reserved IP range set later in BOSH Director. The reserved range is typically the first .1 through .9 addresses in the CIDR range you set for the pcf-infrastructure-subnet-GCP-REGION subnet.
- The IP address cannot be the same as the Gateway IP address set later in Ops Manager. The Gateway IP address is typically the first .1 address in the CIDR range you set for the pcf-infrastructure-subnet-GCP-REGION subnet.

f. For External IP, select Ephemeral.

![Screenshot of network interface]

Note: If you select a static external IP address for the NAT instance, then you can use the static IP to further secure access to your CloudSQL instances.
g. Set IP forwarding to On.
h. Click Done.

6. Click Create to finish creating the NAT instance.

7. Repeat steps 2-6 to create two additional NAT instances with the names and zones specified in the table below. The rest of the configuration remains the same.

<table>
<thead>
<tr>
<th>Instance 2</th>
<th>Name</th>
<th>Select the second zone from your region. Example: For region {us-west1}, select zone {us-west1-b}.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal IP</td>
<td>Custom</td>
<td>and enter an IP address in the Internal IP address field. Example: 192.168.101.3. As described above, this address must in the CIDR range you set for the {pcf-infrastructure-subnet-GCP-REGION} subnet, must exist in a reserved IP range set later in BOSH Director, and cannot be the same as the Gateway IP address set later in Ops Manager.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance 3</th>
<th>Name</th>
<th>Select the third zone from your region. Example: For region {us-west1}, select zone {us-west1-c}.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal IP</td>
<td>Custom</td>
<td>and enter an IP address in the Internal IP address field. Example: 192.168.101.4. As described above, this address must in the CIDR range you set for the {pcf-infrastructure-subnet-GCP-REGION} subnet, must exist in a reserved IP range set later in BOSH Director, and cannot be the same as the Gateway IP address set later in Ops Manager.</td>
</tr>
</tbody>
</table>

Create Routes for NAT Instances

1. Navigate to VPC Networks > Routes.

2. Click CREATE ROUTE.

3. Complete the form as follows:
   - Name: pcf-nat-pri
   - Network: pcf-virt-net
   - Destination IP range: 0.0.0.0/0
   - Priority: 800
   - Instance tags: pcf
   - Next hop: Specify an instance
   - Next hop instance: pcf-nat-gateway-pri

4. Click Create to finish creating the route.

5. Repeat steps 2-4 to create two additional routes with the names and next hop instances specified in the table below. The rest of the configuration remains the same.

<table>
<thead>
<tr>
<th>Route 2</th>
<th>Name: pcf-nat-sec</th>
<th>Next hop instance: pcf-nat-gateway-sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route 3</td>
<td>Name: pcf-nat-ter</td>
<td>Next hop instance: pcf-nat-gateway-ter</td>
</tr>
</tbody>
</table>
Step 5: Create Firewall Rules for the Network

GCP lets you assign tags to VM instances and create firewall rules that apply to VMs based on their tags. For more information about tags, see Labeling Resources in the Google Cloud documentation. This step assigns tags and firewall rules to Ops Manager components and VMs that handle incoming traffic.

1. With your single project or shared-VPC host project selected, navigate to the Networking > VPC network pane and select Firewall rules.

2. Apply the firewall rules in the following table:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Firewall Rules</th>
</tr>
</thead>
</table>
| Rule 1 | This rule allows SSH from public networks.  
Name: `pcf-allow-ssh`  
Network: `pcf-virt-net`  
Allowed protocols and ports: `tcp:22`  
Source filter: IP ranges  
Source IP ranges: `0.0.0.0/0`  
Target tags: `allow-ssh` |
| Rule 2 | This rule allows HTTP from public networks.  
Name: `pcf-allow-http`  
Network: `pcf-virt-net`  
Allowed protocols and ports: `tcp:80`  
Source filter: IP ranges  
Source IP ranges: `0.0.0.0/0`  
Target tags: `allow-http`, `router` |
| Rule 3 | This rule allows HTTPS from public networks.  
Name: `pcf-allow-https`  
Network: `pcf-virt-net`  
Allowed protocols and ports: `tcp:443`  
Source filter: IP ranges  
Source IP ranges: `0.0.0.0/0`  
Target tags: `allow-https`, `router` |
| Rule 4 | This rule allows Gorouter health checks.  
Name: `pcf-allow-http-8080`  
Network: `pcf-virt-net`  
Allowed protocols and ports: `tcp:8080`  
Source filter: IP ranges  
Source IP Ranges: `0.0.0.0/0`  
Target tags: `router` |
| Rule 5 | This rule allows communication between BOSH-deployed jobs.  
Name: `pcf-allow-pas-all`  
Network: `pcf-virt-net`  
Allowed protocols and ports: `tcp;udp;icmp`  
Source filter: Source tags  
Target tags: `pcf`, `pcf-opsman`, `nat-traverse`  
Source tags: `pcf`, `pcf-opsman`, `nat-traverse` |
| Rule 6 (Optional) | This rule allows access to the TCP router.  
Name: `pcf-allow-cf-tcp`  
Network: `pcf-virt-net`  
Source filter: IP ranges  
Source IP ranges: `0.0.0.0/0`  
Allowed protocols and ports: `tcp:1024-65535`  
Target tags: `pcf-cf-tcp` |
|  | This rule allows access to the SSH proxy.  
Name: `pcf-allow-ssh-proxy` |
Rule 7 (Optional)

<table>
<thead>
<tr>
<th>Network</th>
<th>pcf-virt-net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source filter</td>
<td>IP ranges</td>
</tr>
<tr>
<td>Source IP ranges</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>Allowed protocols</td>
<td>tcp:2222</td>
</tr>
<tr>
<td>Target tags</td>
<td>pcf-ssh-proxy, diego-brain</td>
</tr>
</tbody>
</table>

**Note:** If you want your firewall rules to only permit traffic within your private network, modify the **Source IP Ranges** from the table accordingly.

3. If you are only using your GCP project to deploy Ops Manager, then you can delete the following default firewall rules:

- default-allow-http
- default-allow-https
- default-allow-icmp
- default-allow-internal
- default-allow-rdp
- default-allow-ssh

If you are deploying **PAS or other runtimes**, proceed to the following step.

Step 6: Create Database Instance and Databases

Create Database Instance

1. For a shared-VPC installation, select the service project in the GCP console. This step and the following steps allocate resources to the service project, not the host project.

2. From the GCP console, select SQL and click **CREATE INSTANCE**.

3. Ensure **MySQL** is selected and click **Next**.

4. Under **MySQL**, select instance type **Second Generation**.

5. Click **Configure MySQL** under your choice for instance type: Development, Staging, or Production.

6. Configure the instance as follows:

   - **Instance ID**: pcf-pas-sql
   - **Root password**: Set a password for the root user.
   - **Region**: Select the region you specified when creating networks.
   - **Zone**: Any.
   - **Configure machine type and storage**
     - Click **Change** and then select **db-n1-standard-2**.
     - Ensure that **Enable automatic storage increases** is selected. This allows DB storage to grow automatically when space is required.
   - **Enable auto backups and high availability**: Make the following selections:
     - Leave **Automate backups** and **Enable binary logging** selected.
     - Under **High availability**, select the **Create failover replica** checkbox.
   - **Authorize Networks**: Click **Add network** and create a network named **all** that allows traffic from **0.0.0.0/0**.

   **Note:** If you assigned static IP addresses to your NAT instances, you can instead limit access to the database instances by specifying the NAT IP addresses.

7. Click **Create**.

Create Databases
1. Navigate to the **Instances** page and select the database instance you just created.

2. Select the **Databases** tab.

3. Click **Create database** to create the following databases:
   - account
   - app_usage_service
   - autoscale
   - ccdb
   - console
   - diego
   - locket
   - networkpolicyserver
   - nfsvolume
   - notifications
   - routing
   - silk
   - uaa
   - credhub

4. Select the **USERS** tab.

5. Click **Create user account** to create a unique username and password for each database you created above. For **Host name**, select **Allow any host**. You must create a total of fourteen user accounts.

   ![Note: Ensure that the networkpolicyserver database user has the ALL PRIVILEGES permission.]

**Step 7: Create Storage Buckets**

1. With your single project or shared-VPC service project selected in the GCP console, select **Storage > Browser**.

2. Using **CREATE BUCKET**, create buckets with the following names. For **Default storage class**, select **Multi-Regional**:
   - pcf-buildpacks
   - pcf-droplets
   - pcf-packages
   - pcf-resources

**Step 8: Create HTTP Load Balancer**

For load balancing, you can use a global HTTP load balancer or an internal, regional load balancer with a private IP address.

Single-project, standalone installations typically use a global HTTP load balancer. See [Create HTTP Load Balancer](#) for how to set this up.

Shared-VPC installation typically use an internal TCP/UDP load balancer to minimize public IP addresses. See [Create Internal Load Balancer](#) for how to set this up.

**Create Internal Load Balancer**

To create an internal load balancer for Ops Manager on GCP, do the following.

1. Create an internal-facing TCP/UDP load balancer for each region of your PCF deployment.

   ![Note: GCP Internal Load Balancer (iLB) is a regional product. Within the same VPC/network, client VMs in a different region from the iLB cannot access the iLB. See Global Routing Issue in the Google Cloud Setting Up Internal Load Balancing documentation.]

2. Assign private IP addresses to the load balancers.
3. After you have deployed Ops Manager, follow instructions in Create or Update a VM Extension to add a custom VM extension that applies internal load balancing to all VMs deployed by BOSH.

   - For example, the following manifest code adds a VM extension `backend-pool` to PCF VMs:

   ```yaml
   vm_extensions:
     - name: backend-pool
       cloud_properties:
         ephemeral_external_ip: true
         backend_service:
           name: name-of-backend-service
           scheme: INTERNAL
   ```

### Create HTTP Load Balancer

To create a global HTTP load balancer for PCF on GCP, do the following:

1. Create Instance Group
2. Create Health Check
3. Configure Back End
4. Configure Front End

#### Create Instance Group

1. Navigate to Compute Engine > Instance groups.
2. Click CREATE INSTANCE GROUP.
3. Complete the form as follows:
   - For **Name**, enter `pcf-http-lb`
   - For **Location**, select Single-zone.
   - For **Zone**, select the first zone from your region. Example: For region `us-west1`, select zone `us-west1-a`.
   - Under **Group type**, select Unmanaged instance group.
   - For **Network**, select `pcf-virt-net`.
   - For **Subnetwork**, select the `pcf-pas-subnet-my-gcp-region` subnet that you created previously.
   - Click **Create**.

4. Create a second instance group with the following details:
   - **Name**: `pcf-http-lb`
   - **Location**: Single-zone
   - **Zone**: Select the second zone from your region. Example: For region `us-west1`, select zone `us-west1-b`.
   - **Group type**: Select Unmanaged instance group.
   - **Network**: Select `pcf-virt-net`.
   - **Subnetwork**: Select the `pcf-pas-subnet-my-gcp-region` subnet that you created previously.

5. Create a third instance group with the following details:
   - **Name**: `pcf-http-lb`
   - **Location**: Single-zone
   - **Zone**: Select the third zone from your region. Example: For region `us-west1`, select zone `us-west1-c`.
   - **Group type**: Select Unmanaged instance group.
   - **Network**: Select `pcf-virt-net`.
   - **Subnetwork**: Select the `pcf-pas-subnet-my-gcp-region` subnet that you created previously.

#### Create Health Check
1. Navigate to Compute Engine > Health checks.

2. Click CREATE HEALTH CHECK.

3. Complete the form as follows:
   - Name: `pcf-cf-public`
   - Port: 8080
   - Request path: `/health`
   - Check interval: 30
   - Timeout: 5
   - Healthy threshold: 10
   - Unhealthy threshold: 2

4. Click Create.

Configure Back End

1. Navigate to Network services > Load balancing.

2. Click CREATE LOAD BALANCER.

3. Under HTTP(S) Load Balancing, click Start configuration.

4. For the Name, enter `pcf-global-pcf`.

5. Select Backend configuration
   a. From the dropdown, select Backend services > Create a backend service.
   b. Complete the form as follows:
   c. Name: `pcf-http-lb-backend`
   d. Protocol: HTTP
   e. Named port: `http`
   f. Timeout: 10 seconds
   g. Under Backends > New backend, select the Instance group that corresponds to the first zone of the multi-zone instance group you created. For example: `pcf-http-lb (us-west1-a)`. Click Done.
   h. Click Add backend, select the Instance group that corresponds to the second zone of the multi-zone instance group you created. For example: `pcf-http-lb (us-west1-b)`. Click Done.
   i. Click Add backend, select the Instance group that corresponds to the third zone of the multi-zone instance group you created. For example: `pcf-http-lb (us-west1-c)`. Click Done.
   j. Health check: Select the `pcf-cf-public` health check that you created.
   k. Cloud CDN: Ensure Cloud CDN is disabled.
   l. Click Create.

Configure Front End

1. Click Host and path rules to populate the default fields and a green check mark.

2. Select Frontend configuration, and add the following:
   - Name: `pcf-cf-lb-http`
   - Protocol: HTTP
   - IP: Perform the following steps:
     1. Select Create IP address.
     2. Enter a Name for the new static IP address and an optional description. For example, `pcf-global-pcf`.
     3. Click Reserve.
   - Port: 80

3. Click Add Frontend IP and port and add the following:

   Note: Skip this step if you do not have either a self-signed or trusted SSL certificate. When you configure the tile for your chosen runtime, you are given the opportunity to create a new self-signed certificate. Upon creating a certificate, you can complete the Add Frontend IP and port section.
Step 9: Create TCP WebSockets Load Balancer

The load balancer for tailing logs with WebSockets for PCF on GCP operates on TCP port 443.

1. From the GCP console, select Network services > Load balancing > Create load balancer

2. Under TCP Load Balancing, click Start configuration.
3. In the Create a load balancer configuration screen, make the following selections:
   - Under Internet facing or internal only, select From Internet to my VMs.
   - Under Multiple regions or single region, select Single region only.

4. Click Continue.

5. In the New TCP load balancer window, enter pcf-wss-logs in the Name field.

6. Click Backend configuration to configure the Backend service:
Region: Select the region you used to create the network in Create a GCP Network with Subnets.

From the Health check dropdown, create a health check with the following details:

- Name: pcf-gorouter
- Port: 8080
- Request path: /health
- Check interval: 5s
- Timeout: 5s
- Healthy threshold: 10
- Unhealthy threshold: 2

The Backend configuration section shows a green check mark.

7. Click Frontend configuration to open its configuration window and complete the fields:

- Protocol: TCP
- IP: Perform the following steps:
  1. Select Create IP address.
  2. For name Name for the new static IP address and an optional description. For example, pcf-gorouter-wss.
  3. Click Reserve.
8. Click **Review and finalize** to verify your configuration.

9. Click **Create**.

**Step 10: Create SSH Proxy Load Balancer**

1. From the GCP console, select **Network services > Load balancing > Create load balancer**

2. Under **TCP Load Balancing**, click **Start configuration**.

3. Under **Internet facing or internal only**, select **From Internet to my VMs**.

4. Under **Connection termination**, select **No (TCP)**.

5. Click **Continue**.

6. In the **New TCP load balancer** window, enter `pcf-ssh-proxy` in the **Name** field.

7. Select **Backend configuration**, and enter the following field values:
   - **Region**: Select the region you used to create the network in [Create a GCP Network with Subnet](#).
   - **Backup pool**: None
   - **Failover ratio**: 10%
   - **Health check**: No health check

---

© Copyright Pivotal Software Inc, 2013-2019
8. Select **Frontend configuration**, and add the following:

- **Protocol**: TCP
- **IP**: Perform the following steps:
  1. Select **Create IP address**.
  2. Enter a **Name** for the new static IP address and an optional description. For example, `pcf-ssh-proxy`.
  3. Click **Reserve**.
- **Port**: 2222

9. (Optional) Review and finalize your load balancer.

10. Click **Create**.

**Step 11: Create Load Balancer for TCP Router**
To create a load balancer for TCP routing in GCP, do the following:

1. From the GCP console, select Network services > Load balancing > Create load balancer.
2. Under TCP Load Balancing, click Start configuration.
3. Under Connection termination, select No (TCP). Click Continue.

4. On the New TCP load balancer screen, enter a unique name for the load balancer in the Name field. For example, pcf-cf-tcp-lb.
5. Select Backend configuration, and enter the following field values:
   - **Region**: Select the region you used to create the network in Create a GCP Network with Subnet.
   - From the Health check dropdown, create a health check with the following details:
     - **Name**: pcf-tcp-lb
     - **Port**: 80
     - **Request path**: /health
     - **Check interval**: 30
     - **Timeout**: 5
     - **Healthy threshold**: 10
     - **Unhealthy threshold**: 2
     - Click Save and continue.

**Note:** This step is optional and only required if you enable TCP routing in your deployment.
6. Select Frontend configuration, and add the front end IP and port entry as follows:

- Protocol: [TCP]
- IP: Perform the following steps:
  1. Select Create IP address.
  2. Enter a Name for the new static IP address and an optional description. For example, pcf-cf-tcp-lb.
  3. Click Reserve.
- Port: [1024-65535]
7. Click Review and finalize to verify your configuration.

8. Click Create.

**Step 12: Add DNS Records for Your Load Balancers**

In this step, you redirect queries for your domain to the IP addresses of your load balancers.

1. Locate the static IP addresses of the load balancers you created in Preparing to Deploy Ops Manager on GCP Manually:
   - An HTTP(S) load balancer named `pcf-global-pcf`
   - A TCP load balancer for WebSockets named `pcf-wss-logs`
   - A TCP load balancer named `pcf-ssh-proxy`
   - A TCP load balancer named `pcf-cf-tcp-lb`

2. Log in to the DNS registrar that hosts your domain. Examples of DNS registrars include Network Solutions, GoDaddy, and Register.com.

3. Create A records with your DNS registrar that map domain names to the public static IP addresses of the load balancers located above:

<table>
<thead>
<tr>
<th>Create and map this record…</th>
<th>To the IP of this load balancer</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>*.sys.DOMAIN</code> Example: <code>*.sys.example.com</code></td>
<td><code>pcf-global-pcf</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>*.apps.sys.DOMAIN</code> Example: <code>*.apps.example.com</code></td>
<td><code>pcf-global-pcf</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>doppler.sys.DOMAIN</code> Example: <code>doppler.sys.example.com</code></td>
<td><code>pcf-wss-logs</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>loggregator.sys.DOMAIN</code> Example: <code>loggregator.sys.example.com</code></td>
<td><code>pcf-wss-logs</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>ssh.sys.DOMAIN</code> Example: <code>ssh.sys.example.com</code></td>
<td><code>pcf-ssh-proxy</code></td>
<td>Yes, to allow SSH access to apps</td>
</tr>
<tr>
<td><code>tcp.DOMAIN</code> Example: <code>tcp.example.com</code></td>
<td><code>pcf-cf-tcp-lb</code></td>
<td>No, only set up if you have enabled the TCP routing feature</td>
</tr>
</tbody>
</table>

4. Save changes within the web interface of your DNS registrar.
5. In a terminal window, run the following `dig` command to confirm that you created your A record successfully:

```
dig SUBDOMAIN.EXAMPLE-URL.com
```

Where `SUBDOMAIN.EXAMPLE-URL` is the subdomain for your load balancer.

You should see the A record that you just created:

```
;; ANSWER SECTION:
xyz.EXAMPLE.COM.	1767	IN	A	203.0.113.1
```

Next Steps

- (Optional) To prepare for deploying either a PAS or tile on GCP, you can download the needed runtime tile in advance:
  - To download PAS, log in to Pivotal Network, select your desired release version, and download Pivotal Application Service.
  - To download PAS, log in to Pivotal Network, select your desired release version, and download Pivotal Container Service.
  - After initiating the tile download, proceed to the next step, Deploying Ops Manager on GCP Manually.
Deploying Ops Manager on GCP Manually

Page last updated:

This topic describes how to deploy Ops Manager for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

Before you deploy Ops Manager, see the preparation steps in Preparing to Deploy Ops Manager on GCP Manually.

After you complete this procedure, follow the instructions in Configuring BOSH Director on GCP Manually.

Step 1: Locate the Ops Manager Installation File

1. Log in to the Pivotal Network, and click Pivotal Cloud Foundry Operations Manager.

2. From the Releases dropdown, select the release to install.

3. Select one of the following download files:
   - Pivotal Cloud Foundry Ops Manager for GCP
   - Pivotal Cloud Foundry Ops Manager YAML for GCP

   When you click the download link, your browser downloads or opens the OpsManager_VERSION_onGCP.pdf or OpsManager_VERSION_onGCP.yml file.

   These documents provide the GCP location of the Ops Manager.tar.gz installation file based on the geographic location of your installation.

4. Copy the filepath string of the Ops Manager image based on your deployment location.

Step 2: Create a Private VM Image

1. Log in to the GCP console.

2. In the left navigation panel, click Compute Engine, and select Images.

3. Click Create Image.

4. Complete the following fields:
   - **Name**: Enter a name. For example, om-pcf.
   - **Encryption**: Leave Automatic (recommended) selected.
   - **Source**: Choose Cloud Storage file.
   - **Cloud Storage file**: Paste in the Google Cloud Storage filepath you copied from the PDF file in the previous step.
5. Click Create. The file might take a few minutes to import.

Step 3: Create the Ops Manager VM Instance

1. Select the checkbox for the image that you created above.

2. Click Create Instance.

3. In the Create an instance form, complete the following fields:
   - **Name**: Enter a name that matches the naming conventions of your deployment.
   - **Zone**: Choose a zone from the region in which you created your network.
   - **Machine type**: Choose `n1-standard-2`.
   - Click Customize to manually configure the vCPU and memory. An Ops Manager VM instance requires the following minimum specifications:
     - **CPU**: 2 vCPUs
     - **Memory**: 8 GB
   - **Boot disk**: Click Change, then perform the following steps:
     - Click Custom images if it is not already selected.
     - Select the **Boot disk type**. If your environment has high performance needs, select SSD. As an example, environments used to develop PCF tiles might benefit from a higher performing Ops Manager VM boot disk. For most environments, you can select Standard.
     - Set the **Size (GB)** of the boot disk to the minimum or higher.
     - Select the Ops Manager image you created in the previous step if it is not already selected.
Click Select to save.

- Under **Identity and API access**, select the **Service account** you created when preparing your environment during the step [Set up IAM Service Accounts](#).
- Allow **HTTP traffic**: Leave this checkbox unselected.
- Allow **HTTPS traffic**: Leave this checkbox unselected.
- **Networking**: Select the **Networking** tab, and perform the following steps:
  
  - Under **Network interfaces**, perform the following steps:
    - Remove the **default** network interface if this interface still exists.
    - Select the network you created when preparing your environment in the [Create a GCP Network with Subnet](#) section of the Preparing to **Deploy Ops Manager on GCP Manually** topic. For example, `pcf-virt-net`.
    - Under **Subnetwork**, select the `pcf-infrastructure-subnet-GCP-REGION` subnet that you created when preparing your environment in the [Create a GCP Network with Subnet](#) section of the Preparing to **Deploy Ops Manager on GCP Manually** topic.
    - For **Primary internal IP**, select **Ephemeral (Custom)**. Enter an IP address (for example, `192.168.101.5`) in the **Custom ephemeral IP address** field. Specify the next available internal IP address located within the reserved IP range that you configure later in BOSH Director. Do not use the **Gateway IP**, for example, `192.168.101.1`.
    - For **External IP**, select **Create IP address**. In the next form, enter a name for the static IP. For example, `om-public-ip`. Click **Reserve**. In the **External IP** dropdown, select the static IP address you just reserved.
    - For **Network tags**, enter `pcf-opsman` and `allow-https`. These tags apply the firewall rules you created in [Create Firewall Rules for the Network](#) to the Ops Manager VM.

4. Click **Create** to deploy the new Ops Manager VM. This might take a few moments.

5. Navigate to your DNS provider, and create an entry that points a fully qualified domain name (FQDN) `opsman.MY-DOMAIN` to the `pcf-opsman` static IP address of Ops Manager that you created in a previous step.

   **Note**: In order to set up Ops Manager authentication correctly, Pivotal recommends using an FQDN to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

### Next Steps

After you complete this procedure, follow the instructions in the Configuring BOSH Director on GCP Manually topic.

Later on, if you need to SSH into the Ops Manager VM to perform diagnostic troubleshooting, see [SSH into Ops Manager](#).
Configuring BOSH Director on GCP Manually

This topic describes how to configure the BOSH Director in Ops Manager for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

Note: You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain name (FQDN) of Ops Manager that you set up in Deploying Ops Manager on GCP Manually.

Note: In order to set up Ops Manager authentication correctly, Pivotal recommends using an FQDN to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

2. When Ops Manager starts for the first time, you must choose one of the following:
   - **Internal Authentication**: If you use internal authentication, Ops Manager maintains your user database.
   - **SAML Identity Provider**: If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
   - **LDAP Server**: If you use an LDAP server, an external identity server maintains your user database.

Internal Authentication

1. When redirected to the Internal Authentication page, do the following:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI 1.0 topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.
2. Log in to Ops Manager with the Admin username and password that you created in the previous step.

SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.
3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - 6b. BOSH Director SAML service provider metadata: [https://BOSH-IP-ADDRESS:8443/saml/metadata](https://BOSH-IP-ADDRESS:8443/saml/metadata)

   **Note**: To retrieve your **BOSH-IP-ADDRESS**, navigate to the **Status** tab in the **BOSH Director** tile. Record the **BOSH Director IP address**.

7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Audience URI (SP Entity ID)**: [https://OPS-IP:443/uaa](https://OPS-IP:443/uaa)
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on URL**: [https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP](https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP)
   - **Audience URI (SP Entity ID)**: [https://BOSH-IP:8443](https://BOSH-IP:8443)
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

**LDAP Server**

- **For Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - **ldap://**: This specifies that the LDAP server uses an unencrypted connection.
  - **ldaps://**: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or
that you import a trusted certificate to the JVM truststore.

- For **LDAP Username** and **LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN:
  
  \texttt{cn=administrator,ou=Users,dc=example,dc=com}

\begin{itemize}
  \item \textbf{Note:} Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.
\end{itemize}

\begin{itemize}
  \item \textbf{warning:} Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.
\end{itemize}

- For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: \texttt{ou=Users,dc=example,dc=com}

- For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter \texttt{cn=Smith} returns all objects with a common name equal to \texttt{Smith}.

  In the LDAP search filter string that you use to configure your runtime, use \texttt{[0]} instead of the username. For example, use \texttt{cn={0}} to return all LDAP objects with the same common name as the username.

  \begin{itemize}
  \item In addition to \texttt{cn}, other attributes commonly searched for and returned are \texttt{mail}, \texttt{uid}, and, in the case of Active Directory, \texttt{sAMAccountName}.
  \end{itemize}

\begin{itemize}
  \item \textbf{Note:} For instructions for testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry in the Pivotal Support Knowledge Base.
\end{itemize}

- For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

  For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: \texttt{ou=Groups,dc=example,dc=com}

- For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is \texttt{member={0}}.

- For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example \texttt{mail}.

- For **LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.

- From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
  
  - Automatically follow any referrals.
  - Ignore referrals and return partial result.
  - Throw exception for each referral and abort.

- For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

- Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

- If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

- Read the **End User License Agreement**, and select the checkbox to accept the terms.

- Select Provision an admin client in the BOSH UAA. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

- **Click Setup Authentication.**

- **Return to the BOSH Director tile**, and continue with the configuration steps below.
Step 2: Google Cloud Platform Config

1. Click the Google Cloud Platform tile within the Installation Dashboard.

2. Select Google Config. Complete the following fields:
   - **Project ID**: Enter your GCP project ID in lowercase, such as: `your-gcp-project-id`.
   - **Default Deployment Tag**: Enter the prefix that you used when creating the GCP resources for this PCF installation. For example, `pcf`.
   - Select AuthJSON and in the field below enter the contents of the JSON file that you downloaded in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually.

   **Note**: As an alternative, you can select The Ops Manager VM Service Account option to use the service account automatically created by GCP for the Ops Manager VM. To use this option, the project-wide service account that you set up in Set up IAM Service Accounts must be assigned the Service Account Actor role.

3. Click Save.

Step 3: Director Config Page

1. In Ops Manager, select Director Config.
2. In the **NTP Servers (comma delimited)** field, enter *169.254.169.254* to designate the `metadata.google.internal` host as an NTP server.

   *Note: The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify the value of this field.*

3. Leave the **JMX Provider IP Address** field blank.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

   *Note: Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank.*

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the BOSH Resurrector functionality and increase your runtime availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   *Note: If you intend to install, you must enable post-deploy scripts.*

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. **(Optional)** Disable **Allow Legacy Agents** if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.
13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


- **Service Key**: Enter your API service key from PagerDuty.
- **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

- **Host**: Enter your email hostname.
- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in [Preparing CredHub HSMs for Configuration](#).

**Note**: After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
Internal: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.

Luna HSM: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in Preparing CredHub HSMs for Configuration.
3. **Provider Partition Password**
4. **Provider Client Certificate**
5. **Provider Client Certificate Private Key**
6. **HSM Host Address**
7. **HSM Port Address**: If you do not know your port address, enter 1792.
8. **Partition Serial Number**
9. **HSM Certificate**: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. Select a **Blobstore Location** to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable
and less secure, Pivotal recommends that you configure an external blobstore.

**Note:** After you deploy Ops Manager, you cannot change the blobstore location.

### Blobstore Location

- **Internal:** Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.
- **Enable TLS:** Select this checkbox to enable TLS to the blobstore.

  **Note:** If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

- **S3 Compatible Blobstore:** Select this option to use an external S3-compatible endpoint. Follow the procedures in [Sign up for Amazon S3](https://aws.amazon.com/premiumsupport/prime) and [Creating a Bucket](https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-a-bucket.html) in the AWS documentation. When you have created an S3 bucket, complete the following steps:

  1. **S3 Endpoint:** Navigate to the [Regions and Endpoints](https://aws.amazon.com/regions/) topic in the AWS documentation.

     a. Locate the endpoint for your region in the *Amazon Simple Storage Service (S3)* table and construct a URL using your region's endpoint. For example, if you are using the *us-west-2* region, the URL you create would be `https://s3-us-west-2.amazonaws.com`. Enter this URL into the *S3 Endpoint* field.

     b. On a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the

```plaintext
Note: After you deploy Ops Manager, you cannot change the blobstore location.
```
fully qualified domain name of Ops Manager.
c. Copy the custom public CA certificate you used to sign the S3 endpoint into /etc/ssl/certs on the Ops Manager VM.
d. On the Ops Manager VM, run `sudo update-ca-certificates -f` to import the custom CA certificate into the Ops Manager VM truststore.

**Note:** You must also add this custom CA certificate into the Trusted Certificates field in the Security page. See Security Page for instructions.

2. **Bucket Name:** Enter the name of the S3 bucket.
3. **Access Key** and **Secret Key:** Enter the keys you generated when creating your S3 bucket.
4. **Select V2 Signature or V4 Signature.** If you select V4 Signature, enter your Region.

**Note:** AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see Authenticating Requests of in the AWS documentation.

- **GCS Blobstore:** Select this option to use an external Google Cloud Storage (GCS) endpoint. To create a GCS bucket, follow the procedures in Creating Storage Buckets of in the GCS documentation. When you have created a GCS bucket, complete the following steps:
  1. **Bucket Name:** Enter the name of your GCS bucket.
  2. **Storage Class:** Select the storage class for your GCS bucket. See Storage Classes of in the GCP documentation for more information.
  3. **Service Account Key:** Enter the contents of the JSON file that you downloaded in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually.

**Note:** The Pivotal Application Service for Windows (PASW) tile is incompatible with GCP configured with a GCS file store. If you are deploying PASW in your GCP environment, Pivotal recommends that you select the S3-compatible file store for your environment.

18. For **Database Location**, if you configured an external MySQL database such as Cloud SQL, select External MySQL Database and complete the fields below. Otherwise, select Internal. For more information about creating a Cloud SQL instance, see Quickstart for Cloud SQL for MySQL of in the Google Cloud documentation.

- **Host:** Enter the value of your host.
- **Port:** Enter your port number. For example, `3306`.
- **Username:** Enter your username.
- **Password:** Enter your password.
- **Database:** Enter your database name.
  In addition, if you selected the Enable TLS for Director Database checkbox, you can fill out the following optional fields:
  - **Enable TLS:** Selecting this checkbox enables TLS communication between the BOSH Director and the database.
  - **TLS CA:** Enter the Certificate Authority for the TLS Certificate.
  - **TLS Certificate:** Enter the client certificate for mutual TLS connections to the database.
  - **TLS Private Key:** Enter the client private key for mutual TLS connections to the database.
  - **Advanced DB Connection Options:** If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.

19. **(Optional) Modify the Director Workers value,** which sets the number of workers available to execute Director tasks. This field defaults to 5.

20. **(Optional) Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. **(Optional) To add a custom URL for your BOSH Director,** enter a valid hostname in Director Hostname. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director of in the Pivotal Support Knowledge Base.
22. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

23. (Optional) Enter your comma-separated custom **Identification Tags**. For example, `iaas:foundation1,hello:world`. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

24. Click **Save**.

### Step 4: Create Availability Zones Page

- **Note:** Pivotal recommends at least three availability zones for a highly available installation.
  - For configuring a PAS deployment with multiple availability zones, see [Reference Architecture for Pivotal Cloud Foundry on GCP](https://docs.pivotal.io/pcf-gcp-reference-architecture.html).
  - For best practices for distributing workloads across multiple availability zones and clusters, see [Maintaining Workload Uptime](https://docs.pivotal.io/pcf-maintaining-workload-uptime.html).

1. Select **Create Availability Zones**.

2. Click **Add**.

3. For **Google Availability Zone**:
   - Enter one of the zones that you associated to the backend service instance groups of the HTTP(S) Load Balancer. For example, if you are using the `us-central1` region and selected `us-central1-a` as one of the zones for your HTTP(S) Load Balancer instance groups, enter `us-central1-a`.
   - Click **Add**.
   - Repeat the above step for all the availability zones that you associated to instance groups in [Preparing to Deploy Ops Manager on GCP Manually](https://docs.pivotal.io/pcf-preparing-to-deploy-ops-manager-manually.html).

   - Click **Save**.
4. Repeat the above step for all the availability zones you are using in your deployment. When you are done, click **Save**.

## Step 5: Create Networks Page

1. Select **Create Networks**.

2. Make sure **Enable ICMP checks** is not selected. GCP routers do not respond to ICMP pings.

3. Click **Add Network**. Create the three networks described in the tables below.

### Infrastructure Network

<table>
<thead>
<tr>
<th>Network Name</th>
<th><strong>infrastructure</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Network Name</td>
<td>Enter the name of the infrastructure network that you created in the <em>Create a GCP Network with Subnets</em> step of <strong>Preparing to Deploy Ops Manager on GCP Manually</strong>. The format is: <code>pcf-virt-net/pcf-infrastructure-subnet-MY-REGION/MY-GCP-REGION</code></td>
</tr>
<tr>
<td>CIDR</td>
<td>Enter the name of the CIDR ending in <code>/26</code> that you used when you created the infrastructure subnet in GCP. Example: <code>192.168.101.0/26</code></td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>Enter the first <code>.1</code> through <code>.9</code> addresses from the CIDR. For example, if the CIDR is <code>192.168.101.0/26</code>, enter the range <code>192.168.101.1-192.168.101.9</code>.</td>
</tr>
<tr>
<td>DNS</td>
<td><code>169.254.169.254</code></td>
</tr>
<tr>
<td>Gateway</td>
<td>Enter the first <code>.1</code> address from the CIDR. For example, if the CIDR is <code>192.168.101.0/26</code>, enter <code>192.168.101.1</code>.</td>
</tr>
<tr>
<td>Availability Zones</td>
<td>Select all three availability zones.</td>
</tr>
</tbody>
</table>

### Runtime Network

| Name | Enter the name of the runtime that you intend to deploy in this environment. For example, `pas` or `pks`. |
| Google Network Name | Enter the name of the runtime network that you created when you prepared your GCP environment. The format is: `pcf-virt-net/pcf-RUNTIME-subnet-MY-REGION/MY-GCP-REGION` |
| CIDR | Enter the name of the CIDR ending in `/22` that you used when you created the runtime subnet in GCP. Example: `192.168.16.0/22` |
| Reserved IP Ranges | Enter the first `.1` through `.9` addresses from the CIDR. For example, if the CIDR is `192.168.16.0/22`, enter the range `192.168.16.1-192.168.16.9`. |
| DNS | `169.254.169.254` |
| Gateway | Enter the first `.1` address from the CIDR. For example, if the CIDR is `192.168.16.0/22`, enter `192.168.16.1`. |
| Availability Zones | Select all three availability zones. |

### Services Network

| Network Name | **services** |
| Google Network Name | Enter the name of the services network that you created when you prepared your GCP environment. The format is: `pcf-virt-net/pcf-services-subnet-MY-REGION/MY-GCP-REGION` |
| CIDR | Enter the name of the CIDR ending in `/22` that you used when you created the services subnet in GCP. Example: `192.168.20.0/22` |
| Reserved IP Ranges | Enter the first `.1` through `.9` addresses from the CIDR. For example, if the CIDR is `192.168.20.0/22`, enter the range `192.168.20.1-192.168.20.9`. |

---

**Note:** To use a shared VPC network, enter the shared VPC host project name before the network name in the format: `VPC-PROJECT-NAME/NETWORK-NAME/SUBNET-NAME/REGION-NAME`. For example, `vpc-project/opsmgr/central/us-central1`. For more information, see [Configuring a Shared VPC on GCP](#).
<table>
<thead>
<tr>
<th><strong>DNS</strong></th>
<th>169.254.169.254</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gateway</strong></td>
<td>Enter the first .1 address from the CIDR. For example, if the CIDR is 192.168.16.0/22, enter 192.168.16.1.</td>
</tr>
<tr>
<td><strong>Availability Zones</strong></td>
<td>Select all three availability zones.</td>
</tr>
</tbody>
</table>

- **Note:** After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see Expanding Your Network with Additional Subnets.

- **Note:** Ensure 169.254.169.254, the IP address of the metadata.google.internal metadata server, is always specified at the beginning of this step’s DNS resolver lists.

---

### Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.

2. Use the dropdown to select a **Singleton Availability Zone**. The BOSH Director installs in this Availability Zone.

3. Under **Network**, select the **infrastructure** network for your BOSH Director.

4. Click **Save**.

### Step 7: Security Page

1. Select **Security**.

![Security Page](image)

- **Include OpsManager Root CA in Trusted Certs**

- **Generate VM passwords or use single password for all VMs**

- **Generate passwords**

- **Use default BOSH password**

- **Save**

2. In **Trusted Certificates**, enter your custom certificate authority (CA) certificates to insert into your organization’s certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEF0123456789ABCD0123456789
EFGH0123456789ABCD0123456789
-----END CERTIFICATE-----
```
3. (Optional) Select the Include OpsManager Root CA in Trusted Certs checkbox to include the Ops Manager root CA in the Trusted Certificates field. BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose Generate passwords or Use default BOSH password. Pivotal recommends that you use the Generate passwords option for greater security.

5. Click Save. To view your saved Director password, click the Credentials tab.

Step 8: BOSH DNS Config Page

1. Select BOSH DNS Config.

2. (Optional) In Excluded Recursors, enter a list of prohibited recursor addresses.

3. (Optional) In Recursor Timeout, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   Note: This time limit must include one of the Go parse duration time units. For example, entering 5s sets the timeout limit to five seconds. For more information about supported time units, see func ParseDuration in the Go Programming Language documentation.

4. (Optional) In Handlers, enter a list of custom domain handlers in JSON format.

5. Click Save.

Step 9: Syslog Page

© Copyright Pivotal Software Inc, 2013-2019
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.
7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** field. This field requires the rainerscript syntax.

10. Click **Save Syslog Settings**.

**Step 10: Resource Config Page**

1. Select **Resource Config**.

2. Ensure that the **Internet Connected** checkboxes are not selected for any jobs. The checkbox gives VMs a public IP address that enables outbound internet access. In [Preparing to Deploy Ops Manager on GCP Manually](#), you provisioned a Network Address Translation (NAT) box to provide internet connectivity to your VMs. For more information about using NAT in GCP, see the [GCP documentation](#).

3. **Note:** If you install Pivotal Application Service for Windows (PASW), provision your **Master Compilation Job** with at least 100 GB of disk space.

**Step 11: (Optional) Add Custom VM Extensions**

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see [Managing Custom VM Extensions](#).

**Step 12: Complete the BOSH Director Installation**

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Review Pending Changes**, then **Apply Changes**. If the following ICMP error message appears, return to the **Network Config** screen, and ensure you have deselected the **Enable ICMP Checks**. Click **Review Pending Changes**, then **Apply Changes** again.

3. BOSH Director installs. This might take a few moments. When the installation process successfully completes, the **Changes Applied** window appears.

**Note:** Pivotal strongly recommends that you enable TLS encryption when you are forwarding logs. Logs can contain sensitive information, such as cloud provider credentials.
Next Steps

After you complete this procedure, follow the instructions for the runtime you intend to install.

- To deploy PAS, see Configuring PAS.
- To prepare to deploy, see Creating Service Accounts in GCP for GCP.
Installing PCF on GCP Using Terraform

This topic explains how to install PCF on GCP using Terraform.

Overview

You can install PCF on GCP with either the Pivotal Application Service (PAS) or Pivotal Container Service (PKS) runtime.

To install PCF on GCP using Terraform, do one of the following:

- Install with PAS. See Install on PAS Using Terraform.
- Install with PKS. See Install on PKS Using Terraform.

Install with PAS Using Terraform

To install with PAS using Terraform, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on GCP Using Terraform.
2. Configure BOSH Director. See Configuring BOSH Director on GCP Using Terraform.
3. (Optional) Configure a shared VPC. See (Optional) Configuring a Shared VPC on GCP.
4. Configure PAS. See Configuring PAS.

Install with PKS Using Terraform

To install with PKS using Terraform, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on GCP Using Terraform.
2. Configure BOSH Director. See Configuring BOSH Director on GCP Using Terraform.
3. (Optional), Configure a shared VPC. See (Optional) Configuring a Shared VPC on GCP.
4. Configure PKS. See Installing PKS on GCP.
Deploying Ops Manager on GCP Using Terraform

This guide describes the preparation steps required to install Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP) using Terraform templates.

The Terraform template for PCF on GCP describes a set of GCP resources and properties. For more information about how Terraform creates resources in GCP, see the Google Cloud Provider topic on the Terraform site.

You may also find it helpful to review different deployment options in the Reference Architecture for Pivotal Cloud Foundry on GCP.

Prerequisites

In addition to fulfilling the prerequisites listed in the Installing Pivotal Cloud Foundry on GCP topic, ensure you have the following:

- The Terraform CLI
- The Google Cloud SDK
- In your GCP project, enable the following APIs:
  - Identity and Access Management
  - Cloud Resource Manager
  - Cloud DNS
  - Cloud SQL API
  - Compute Engine API

Step 1: Obtain a GCP Service Account Key File

To use the Terraform templates to create the necessary infrastructure resources for PCF, you need a service account key file.

To create an account key file, follow the procedure below corresponding to your own use case.

- I already have a service account I want to use:
  1. Navigate to the GCP console.
  2. Select IAM and locate your service account.
  3. From the Options column, open the dropdown and click Create Key.

- I want to create a new service account:
  1. Open a terminal window.
  2. To create a service account using the gcloud CLI, run the following command:

    ```bash
gcloud iam service-accounts create ACCOUNT-NAME
    ```

  3. To create a key file for your service account, run the following command:

    ```bash
gcloud iam service-accounts keys create "terraform.key.json" --iam-account "ACCOUNT-NAME@PROJECT-ID.iam.gserviceaccount.com"
    ```

  4. To bind the service account to your project and give it the owner role, run the following command:

    ```bash
gcloud projects add-iam-policy-binding PROJECT-ID --member "serviceAccount:ACCOUNT-NAME@PROJECT-ID.iam.gserviceaccount.com" --role 'roles/owner'
    ```

Where:

- `ACCOUNT-NAME` is the name you want to apply to the new account.
- `PROJECT-ID` is your Google Cloud Platform Project ID.
Step 2: Download Templates and Edit Variables File

Before you can run Terraform commands to provision infrastructure resources, you must download the GCP Terraform Templates and create a Terraform template variables file as described below:

1. On Pivotal Network, navigate to the Pivotal Application Service (PAS) release.
2. Download the GCP Terraform file.
3. Extract the contents of the ZIP file.
4. Move the extracted folder to the `workspace` directory on your local machine.
5. On the command line, navigate to the directory. For example:

   ```
   $ cd ~/workspace/pivotal-cf-terraforming-gcp
   ```

6. Navigate to the `terraforming-pas` or `terraforming-pks` directory that contains the Terraform files for your runtime.
7. In the runtime directory, create a text file named `terraform.tfvars`.
8. Open the `terraform.tfvars` file and add the following:

   ```
   env_name = "YOUR-ENVIRONMENT-NAME"
   opsman_image_url = "YOUR-OPS-MAN-IMAGE-URL"
   region = "YOUR-GCP-REGION"
   zones = ["YOUR-AZ-1", "YOUR-AZ-2", "YOUR-AZ-3"]
   project = "YOUR-GCP-PROJECT"
   dns_suffix = "YOUR-DNS-SUFFIX"
   ssl_cert = <<SSL_CERT
               -----BEGIN CERTIFICATE-----
               YOUR-CERTIFICATE
               -----END CERTIFICATE-----
               SSL_CERT
   ssl_private_key = <<SSL_KEY
                     -----BEGIN EXAMPLE RSA PRIVATE KEY-----
                     YOUR-PRIVATE-KEY
                     -----END EXAMPLE RSA PRIVATE KEY-----
                     SSL_KEY
   service_account_key = "SERVICE_ACCOUNT_KEY"
   YOUR-KEY-JSON
   SERVICE_ACCOUNT_KEY
   ```

9. Edit the values in the file according to the table below.

<table>
<thead>
<tr>
<th>Value to replace</th>
<th>Guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOUR-ENVIRONMENT-NAME</td>
<td>Enter a name to use to identify resources in GCP. Terraform prepends the names of the resources it creates with this environment name. Example: <code>pcf</code>.</td>
</tr>
<tr>
<td>YOUR-OPS-MAN-IMAGE-URL</td>
<td>Enter the source URL of the Ops Manager image you want to boot. You can find this URL in the PDF included with the Ops Manager release on Pivotal Network.</td>
</tr>
<tr>
<td>YOUR-GCP-REGION</td>
<td>Enter the name of the GCP region in which you want Terraform to create resources. Example: <code>us-central1</code>.</td>
</tr>
<tr>
<td>YOUR-AZ-1, YOUR-AZ-2,</td>
<td>Enter three availability zones from your region. Example: <code>us-central1-a, us-central1-b, us-central1-c</code>.</td>
</tr>
<tr>
<td>YOUR-AZ-3</td>
<td></td>
</tr>
</tbody>
</table>
   | YOUR-GCP-PROJECT       | Enter the name of the GCP project in which you want Terraform to create resources. Enter a domain name to use as part of the system domain for your PCF deployment. Terraform creates DNS records in GCP using `YOUR-ENVIRONMENT-NAME` and `YOUR-DNS-SUFFIX`. For example, if you enter `example.com` for your DNS suffix and have `pcf` as your environment name, Terraform creates DNS records at `pcf.example.com`. Enter a certificate to use for HTTP load balancing. For production environments, use a certificate from a trusted certificate authority. Enter your certificate as a string. For example:

   ```
   ---BEGIN CERTIFICATE-----
   YOUR-CERTIFICATE
   -----END CERTIFICATE-----
   ```

   Enter your certificate as a string. For example:

   ```
   ---BEGIN RSA PRIVATE KEY-----
   YOUR-PRIVATE-KEY
   -----END RSA PRIVATE KEY-----
   ```

   Enter your certificate as a string. For example:

   ```
   -----BEGIN SERVICE ACCOUNT KEY-----
   YOUR-KEY-JSON
   -----END SERVICE ACCOUNT KEY-----
   ```

© Copyright Pivotal Software Inc, 2013-2019
Step 3: Add Optional Variables

Complete this step if you want to do any of the following:

- Change the default CIDR ranges
- Deploy the Isolation Segment tile
- Use an external Google Cloud SQL database
- Use external Google Storage buckets
- Disable generated GCP service account key for blobstore

In your `terraform.tfvars` file, specify the appropriate variables from the sections below.

**Note:** You can see the configurable options by opening the `variables.tf` file and looking for variables with default values.

### CIDR Ranges for Subnets

If you want to change the CIDR ranges for the infrastructure, your runtime, or services networks that Terraform creates, add the following variables to your `terraform.tfvars` file, replacing `YOUR-INFRASTRUCTURE-CIDR`, `YOUR-RUNTIME-CIDR`, and `YOUR-SERVICES-CIDR` with your desired values.

```plaintext
infrastructure_cidr = YOUR-INFRASTRUCTURE-CIDR
pas_cidr = YOUR-RUNTIME-CIDR
services_cidr = YOUR-SERVICES-CIDR
```

### Isolation Segments

If you plan to deploy the Isolation Segment tile, add the following variables to your `terraform.tfvars` file, replacing `YOUR-CERTIFICATE`, `YOUR-PRIVATE-KEY`, and `YOUR-CERTIFICATE` with a certificate and private key. This causes Terraform to create an additional HTTP load balancer across three availability zones to use for the Isolation Segment tile.

```plaintext
isolation_segment = true
iso_seg_ssl_cert = <<ISO_SEG_SSL_CERT
    -----BEGIN CERTIFICATE-----
    YOUR-CERTIFICATE
    -----END CERTIFICATE-----
iso_seg_ssl_cert_private_key = <<ISO_SEG_SSL_KEY
    -----BEGIN EXAMPLE RSA PRIVATE KEY-----
    YOUR-PRIVATE-KEY
    -----END EXAMPLE RSA PRIVATE KEY-----
```

### External Database

1. If you want to use an external Google Cloud SQL database for Ops Manager and Pivotal Application Service (PAS), add the following to your `terraform.tfvars` file:

```plaintext
external_database = true
```
2. If you want to specify a single host from which users can connect to the Ops Manager and runtime databases, add the following to your `terraform.tfvars` file:

```terraform
opsman_sql_db_host = HOST-IP-ADDRESS
pas_sql_db_host = HOST-IP-ADDRESS
```

Where `HOST-IP-ADDRESS` is your desired IP address(es).

**External Storage Buckets**

If you want to use Google Cloud Storage buckets for the PAS Cloud Controller, add the following to your `terraform.tfvars` file:

```terraform
create_gcs_buckets = true
```

**GCP Service Account Key for Blobstore**

If you want to provide your own service account for blob storage instead of using a generated service account, add the following to your `terraform.tfvars` file:

```terraform
create_blobstore_service_account_key = false
```

**Step 4: Create GCP Resources with Terraform**

Follow these steps to use the Terraform CLI to create resources on GCP:

1. From the directory that contains the Terraform files, run the following command to initialize the directory based on the information you specified in the `terraform.tfvars` file.

   ```sh
terraform init
   ```

2. Run the following command to create the execution plan for Terraform.

   ```sh
terraform plan -out=plan
   ```

3. To execute the plan from the previous step, run the following command:

   ```sh
terraform apply plan
   ```

   *Note:* It may take several minutes for Terraform to create all the resources in GCP.

**Step 5: Create DNS Record**

1. In a browser, navigate to the DNS provider for the DNS suffix you entered in your `terraform.tfvars` file.

2. Create a new NS (Name server) record for your PCF system domain. Your system domain is `YOUR-ENVIRONMENT-NAME.YOUR-DNS-SUFFIX`.

   a. In this record, enter the name servers included in `env_dns_zone_name_servers` from your Terraform output.

**Next Steps**

After you complete this procedure, follow the instructions in the Configuring BOSH Director on GCP Using Terraform topic.
Configuring BOSH Director on GCP Using Terraform

Prerequisite

To complete the procedures in this topic, you must have access to the output from when you ran `terraform apply` to create resources for this deployment. You can view this output at any time by running `terraform output`. You use the values in your Terraform output to configure the BOSH Director tile.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain name (FQDN) of the BOSH Director. Use the `ops_manager_dns` value from running `terraform output`. You can also perform the procedures in this topic using the Ops Manager API. For more information, see the Using the Ops Manager API topic.

   **Note:** In order to set up Ops Manager authentication correctly, Pivotal recommends using a Fully Qualified Domain Name (FQDN) to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access.

1. When Ops Manager starts for the first time, you must choose one of the following:
   - **Internal Authentication:** If you use internal authentication, Ops Manager maintains your user database.
   - **SAML Identity Provider:** If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
   - **LDAP Server:** If you use an LDAP Server, an external identity server maintains your user database.

   ![Welcome to Ops Manager](image)

   **Select an Authentication System**
   - Internal Authentication
   - SAML Identity Provider
   - LDAP Server

   **Upgrading Ops Manager?**
   - Import Existing Installation

   **Internal Authentication**

   1. When redirected to the Internal Authentication page, do the following:
Enter a Username, Password, and Password confirmation to create an Admin user.

Enter a Decryption passphrase and the Decryption passphrase confirmation. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

If you are using an HTTP proxy or HTTPS proxy, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.

Read the End User License Agreement, and select the checkbox to accept the terms.

Click Setup Authentication.

2. Log in to Ops Manager with the Admin username and password you created in the previous step.

SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.
3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - 6a. Ops Manager SAML service provider metadata: https://OPS-MAN-FQDN:443/uaa/saml/metadata
   - 6b. BOSH Director SAML service provider metadata: https://BOSH-IP-ADDRESS:8443/saml/metadata

   **Note:** The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager login page.

7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Single sign on URL**: https://OPS-MAN-FQDN:443/uaa/saml/SSO/alias/OPS-MAN-FQDN
   - **Audience URI (SP Entity ID)**: https://OP-MAN-FQDN:443/uaa
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on URL**: https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP
   - **Audience URI (SP Entity ID)**: https://BOSH-IP:8443
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

**LDAP Server**

- For **Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - ldap://: This specifies that the LDAP server uses an unencrypted connection.
  - ldaps://: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or...
that you import a trusted certificate to the JVM truststore.

- For **LDAP Username** and **LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: cn=administrator,ou=Users,dc=example,dc=com

  **Note:** Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

  **warning:** Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.
  
  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: ou=Users,dc=example,dc=com

- For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to Smith.

  In the LDAP search filter string that you use to configure your runtime, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

  **Note:** Other attributes commonly searched for and returned are `mail`, `uid`, and, in the case of Active Directory, `sAMAccountName`.

- For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

  For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: ou=Groups,dc=example,dc=com

  - For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is `member={0}`.
  - For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example `mail`.
  - For **LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.

  - From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
    - Automatically follow any referrals.
    - Ignore referrals and return partial result.
    - Throw exception for each referral and abort.

- For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

- Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

- If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

- Read the **End User License Agreement**, and select the checkbox to accept the terms.

- Select **Provision an admin client in the BOSH UAA**. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

- Click **Setup Authentication**.

- Return to the **BOSH Director** tile, and continue with the configuration steps below.

© Copyright Pivotal Software Inc, 2013-2019 309
Step 2: Google Cloud Platform Config

1. Click the Google Cloud Platform tile within the Installation Dashboard.

2. Select Google Config. Complete the following fields:
   - **Project ID**: Enter the value of `project` from your `terraform.tfvars` file.
   - **Default Deployment Tag**: Enter the value of `env_name` from your `terraform.tfvars` file.
   - Select AuthJSON and in the field below enter the contents of the JSON file for your service account key.

   **Note**: As an alternative, you can select The Ops Manager VM Service Account option to use the service account automatically created by GCP for the Ops Manager VM.

3. Click Save.

Step 3: Director Config Page

1. In Ops Manager, select Director Config.
2. In the **NTP Servers (comma delimited)** field, enter `169.254.169.254` to designate the `metadata.google.internal` host as NTP server.

   **Note:** The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify the value of this field.

3. Leave the **JMX Provider IP Address** field blank.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

   **Note:** Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the BOSH Resurrector functionality and increase your runtime availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   **Note:** If you intend to install, you must enable post-deploy scripts.

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. *(Optional)* Disable **Allow Legacy Agents** if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.
13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


![HM Pager Duty Plugin](image)

- **Service Key**: Enter your API service key from PagerDuty.
- **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

![HM Email Plugin](image)

- **Host**: Enter your email hostname.
- **Port**: Enter your email port number.
- **Domain**: Enter your domain.
- **From**: Enter the address for the sender.
- **Recipients**: Enter comma-separated addresses of intended recipients.
- **Username**: Enter the username for your email server.
- **Password**: Enter the password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in **Preparing CredHub HSMs for Configuration**.

**Note**: After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
Internal: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.

Luna HSM: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in [Preparing CredHub HSMs for Configuration](#).
3. **Provider Partition Password**
4. **Provider Client Certificate**
5. **Provider Client Certificate Private Key**
6. **HSM Host Address**
7. **HSM Port Address**: If you do not know your port address, enter 1792.
8. **Partition Serial Number**
9. **HSM Certificate**: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. Select a **Blobstore Location** to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable
and less secure, Pivotal recommends that you configure an external blobstore.

Note: After you deploy Ops Manager, you cannot change the blobstore location.

- **Internal**: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.

- **Enable TLS**: Select this checkbox to enable TLS to the blobstore.

Note: If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

- **S3 Compatible Blobstore**: Select this option to use an external S3-compatible endpoint. Follow the procedures in Sign up for Amazon S3 and Creating a Bucket in the AWS documentation. When you have created an S3 bucket, complete the following steps:

  1. **S3 Endpoint**: Navigate to the Regions and Endpoints topic in the AWS documentation.

      a. Locate the endpoint for your region in the Amazon Simple Storage Service (S3) table and construct a URL using your region’s endpoint. For example, if you are using the us-west-2 region, the URL you create would be https://s3-us-west-2.amazonaws.com. Enter this URL into the **S3 Endpoint** field.

      b. On a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the
fully qualified domain name of Ops Manager.

c. Copy the custom public CA certificate you used to sign the S3 endpoint into /etc/ssl/certs on the Ops Manager VM.
d. On the Ops Manager VM, run `sudo update-ca-certificates -f` to import the custom CA certificate into the Ops Manager VM truststore.

**Note:** You must also add this custom CA certificate into the Trusted Certificates field in the Security page. See Security Page for instructions.

2. **Bucket Name:** Enter the name of the S3 bucket.

3. **Access Key** and **Secret Key:** Enter the keys you generated when creating your S3 bucket.

4. **Select V2 Signature or V4 Signature.** If you select V4 Signature, enter your Region.

**Note:** AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see Authenticating Requests in the AWS documentation.

- **GCS Blobstore:** Select this option to use an external Google Cloud Storage (GCS) endpoint. To create a GCS bucket, follow the procedures in Creating Storage Buckets in the GCS documentation. When you have created a GCS bucket, complete the following steps:

  1. **Bucket Name:** Enter the name of your GCS bucket.
  2. **Storage Class:** Select the storage class for your GCS bucket. See Storage Classes in the GCP documentation for more information.
  3. **Service Account Key:** Enter the contents of the JSON file that you downloaded in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually.

**Note:** The Pivotal Application Service for Windows (PASW) tile is incompatible with GCP configured with a GCS file store. If you are deploying PASW in your GCP environment, Pivotal recommends that you select the S3-compatible file store for your environment.

18. For **Database Location**, if you configured your `terraform.tfvars` file to create an external database for Ops Manager, select External MySQL Database and complete the fields below. Otherwise, select Internal.

- **Host:** Enter the value of `sql_db_ip` from your Terraform output.
- **Port:** Enter `3306`.
- **Username:** Enter the value of `opsman_sql_username` from your Terraform output.
- **Password:** Enter the value of `opsman_sql_password` from your Terraform output.
- **Database:** Enter the value of `opsman_sql_db_name` from your Terraform output.

In addition, if you selected External MySQL Database, you can fill out the following optional fields:

- **Enable TLS:** Selecting this checkbox enables TLS communication between the BOSH Director and the database.
- **TLS CA:** Enter the Certificate Authority for the TLS Certificate.
- **TLS Private Key:** Enter the client private key for mutual TLS connections to the database.

**Note:** You must select Enable TLS for Director Database to configure the TLS-related fields.

- **Advanced DB Connection Options:** If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.

19. (Optional) Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to 5.

20. (Optional) **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. (Optional) To add a custom URL for your BOSH Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director in the Pivotal Support Knowledge Base.
22. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

23. (Optional) Enter your comma-separated custom **Identification Tags**. For example, `iaas:foundation1, hello:world`. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

24. Click **Save**.

### Step 4: Create Availability Zones Page

*Note: Pivotal recommends at least three availability zones for a highly available installation.*

- For configuring a PAS deployment with multiple availability zones, see [Reference Architecture for Pivotal Cloud Foundry on GCP](#).
- For best practices for distributing workloads across multiple availability zones and clusters, see [Maintaining Workload Uptime](#).

1. Select **Create Availability Zones**.

2. Use the **Add** button to add three availability zones corresponding to those listed in the `azs` field in your Terraform output.

3. Click **Save**.

### Step 5: Create Networks Page

1. Select **Create Networks**.

2. Make sure **Enable ICMP checks** is not selected. GCP routers do not respond to ICMP pings.

3. Use the **Add Network** button to create the following three networks:

   - **Infrastructure Network**
     - **Network Name**: `Infrastructure`
     - **Google Network Name**: Use the `network_name`, `infrastructure_subnet_name`, and `region` fields from your Terraform output to enter the name of the infrastructure network created by Terraform.
     - **CIDR**: Enter the value of `infrastructure_subnet_cidrs` from your Terraform output.
     - **Reserved IP Ranges**: Enter the first `1-31` through `32-63` addresses from the CIDR. For example, if the CIDR is `192.168.101.0/26`, enter the range `192.168.101.1-192.168.101.9`.

*Note: To use a shared VPC network, enter the shared VPC host project name before the network name in the format `VPC-PROJECT-NAME/NETWORK-NAME/SUBNET-NAME/REGION-NAME`. For example, `vpc-project/opsmgr/central/us-central1`. For more information, see [Configuring a Shared VPC on GCP](#).*
### DNS Gateway
Enter the value of `infrastructure_subnet_gateway` from your Terraform output.

### Availability Zones
Select all three availability zones.

### Runtime Network

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Name</strong></td>
<td>Enter the name of the runtime that you intend to deploy in this environment. For example, <code>pas</code> or <code>pks</code>.</td>
</tr>
<tr>
<td><strong>Google Network Name</strong></td>
<td>Use the <code>network_name</code>, <code>pas_subnet_name</code>, and <code>region</code> fields from your Terraform output to enter the name of the runtime network created by Terraform. The format is: <code>network_name/pas_subnet_name/region</code>.</td>
</tr>
<tr>
<td><strong>CIDR</strong></td>
<td>Enter the value of <code>pas_subnet_cidrs</code> from your Terraform output.</td>
</tr>
<tr>
<td><strong>Reserved IP Ranges</strong></td>
<td>Enter the first <code>.1</code> through <code>.9</code> addresses from the CIDR. For example, if the CIDR is <code>192.168.16.0/22</code>, enter the range <code>192.168.16.1-192.168.16.9</code>.</td>
</tr>
<tr>
<td><strong>DNS</strong></td>
<td>169.254.169.254</td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>Enter the value of <code>pas_subnet_gateway</code> from your Terraform output.</td>
</tr>
<tr>
<td><strong>Availability Zones</strong></td>
<td>Select all three availability zones.</td>
</tr>
</tbody>
</table>

### Services Network

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Name</strong></td>
<td><code>services</code></td>
</tr>
<tr>
<td><strong>Google Network Name</strong></td>
<td>Use the <code>network_name</code>, <code>services_subnet_name</code>, and <code>region</code> fields from your Terraform output to enter the name of the services network created by Terraform. The format is: <code>network_name/services_subnet_name/region</code>.</td>
</tr>
<tr>
<td><strong>CIDR</strong></td>
<td>Enter the value of <code>services_subnet_cidrs</code> from your Terraform output.</td>
</tr>
<tr>
<td><strong>Reserved IP Ranges</strong></td>
<td>Enter the first <code>.1</code> through <code>.9</code> addresses from the CIDR. For example, if the CIDR is <code>192.168.20.0/22</code>, enter the range <code>192.168.20.1-192.168.20.9</code>.</td>
</tr>
<tr>
<td><strong>DNS</strong></td>
<td>169.254.169.254</td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>Enter the value of <code>services_subnet_gateway</code> from your Terraform output.</td>
</tr>
<tr>
<td><strong>Availability Zones</strong></td>
<td>Select all three availability zones.</td>
</tr>
</tbody>
</table>

**Note:** After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see [Expanding Your Network with Additional Subnets](#).

### Step 6: Assign AZs and Networks Page

1. Select **Assign AZs and Networks**.

2. Use the dropdown to select a **Singleton Availability Zone**. The BOSH Director installs in this Availability Zone.

3. Under **Network**, select the **infrastructure** network for your BOSH Director.

4. Click **Save**.

### Step 7: Security Page

1. Select **Security**.
2. In **Trusted Certificates**, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEFGH12345678ABCDEFGH12345678ABCDEFGH12345678ABCDEFGH12345678...
------END CERTIFICATE------
-----BEGIN CERTIFICATE-----
BCDEFGH12345678ABCDEFGH12345678ABCDEFGH12345678...
------END CERTIFICATE------
-----BEGIN CERTIFICATE-----
CDEFGH12345678ABCDEFGH12345678...
------END CERTIFICATE------
```

3. (Optional) Select the **Include OpsManager Root CA in Trusted Certs** checkbox to include the Ops Manager root CA in the Trusted Certificates field. BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

5. Click **Save**. To view your saved Director password, click the **Credentials** tab.

---

**Step 8: Syslog Page**

---

© Copyright Pivotal Software Inc, 2013-2019 318 2.5
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.

   b. Enter the SSL certificate for the remote server in **SSL Certificate**.

   © Copyright Pivotal Software Inc, 2013-2019
7. (Optional) Enter an integer in Queue Size. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to Forward Debug Logs to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the Custom rsyslog Configuration field. This field requires the rainerscript syntax.

10. Click Save Syslog Settings.

**Step 9: Resource Config Page**

1. Select Resource Config.

2. Verify that the Internet Connected checkbox for every job is checked. The Terraform templates do not provision a Network Address Translation (NAT) box for internet connectivity to your VMs, so instead VMs are provided with ephemeral public IP addresses to allow the jobs to reach the internet.

   **Note:** If you want to provision a Network Address Translation (NAT) box to provide internet connectivity to your VMs instead of providing them with public IP addresses, deselect the Internet Connected checkboxes. For more information about using NAT in GCP, see the [GCP documentation](https://cloud.google.com/).

   **Note:** If you install Pivotal Application Service for Windows (PASW), provision your Master Compilation Job with at least 100 GB of disk space.

**Step 10: (Optional) Add Custom VM Extensions**

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see [Managing Custom VM Extensions](#).

**Step 11: Complete the BOSH Director Installation**

1. Click the Installation Dashboard link to return to the Installation Dashboard.

2. Click Review Pending Changes, then Apply Changes. If the following ICMP error message appears, return to the Network Config screen, and make sure you have deselected the Enable ICMP Checks box. Click Review Pending Changes, then Apply Changes again.

   Please review the errors below
   - Cannot reach gateway with IP 10.0.11.1 (ignorable if ICMP is disabled)
   - Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)

   ![PCF Ops Manager Dashboard](image)

   **Stop and fix errors**

3. BOSH Director installs. This may take a few moments. When the installation process successfully completes, the Changes Applied window appears.

© Copyright Pivotal Software Inc, 2013-2019
Next Steps

After you complete this procedure, follow the instructions for the runtime you intend to install.

- To deploy PAS, see Configuring PAS.
- To prepare to deploy, see Creating Service Accounts in GCP for.
Configuring a Shared VPC on GCP

Page last updated:

This guide describes the preparation steps required to configure and integrate a shared Virtual Private Cloud (VPC) on Google Cloud Platform (GCP) with Pivotal Cloud Foundry (PCF).

GCP Shared VPC, formerly known as Google Cross-Project Networking (XPN), enables you to assign GCP resources to individual projects within an organization but allows communication and shared services between projects. For more information about shared VPCs, see Shared VPC Overview in the GCP documentation.

Prerequisites

To configure a shared VPC, you must assign your project to a Cloud Organization. Confirm that you have a Cloud Organization associated with your GCP account using one of the following methods:

- **GCP Console**: From <https://console.cloud.google.com>, click the **Organization** dropdown at the top of the page to display all organizations you belong to.
- **gcloud Command Line Interface (CLI)**: From the command line, run `gcloud organizations list` to display all organizations you belong to. See gcloud Overview in the Google documentation to install the gcloud CLI.

For more information, see Creating and Managing Organizations in the GCP documentation. If you do not have a Cloud Organization, contact GCP support.

Step 1: Provision the Shared VPC

Follow the Enabling a shared VPC host project procedure in the GCP documentation. This procedure requires shared VPC admin permissions.

Step 2: Create a Shared VPC Network

Use the procedures in the Preparing to Deploy Ops Manager on GCP Manually topic to create a new network with firewall rules. Do the following:

- **Step 3: Create a GCP Network with Subnet**
- **Step 5: Create Firewall Rules for the Network**

Step 3: Connect the Shared VPC to Ops Manager

You can use the GCP console or the gcloud CLI to connect the shared VPC host project with Ops Manager.

For more information, see VPC Network Peering in the GCP documentation.

⚠️ warning: VPC Network Peering is currently in beta and intended for evaluation and test purposes only.

Set Up VPC Network Peering with GCP Console

To set up VPC network peering with the GCP console, do the following:

1. From the GCP console, click **Networking**, then **VPC network peering**.
2. Click **Create Connection**.

3. Enter a name for the network connection from the Ops Manager project to the new shared network, such as `opsmanager-to-xpn`.

4. Click **Save**.

5. Click **Create Connection**.

6. Enter a name for the network connection from the new shared network to the Ops Manager project, such as `xpn-to-opsmanager`.
Set Up VPC Network Peering with gcloud CLI

To set up VPC network peering with the gcloud CLI, do the following:

1. Enter the following command, replacing `OPSMANAGER-PROJECT` with the name of the project that contains your Ops Manager installation:

   ```
   $ gcloud config set project OPSMANAGER-PROJECT
   ```

2. Enter the following command to create a connection from the Ops Manager project to the new shared VPC project:

   ```
   $ gcloud beta compute networks peerings create OPSMANAGER-TO-VPC
     --network OPSMANAGER-NETWORK
     --peer-project VPC-HOST-PROJECT
     --peer-network VPC-NETWORK
     --auto-create-routes
   ```

   Replace the following text in the command above:
   - `OPSMANAGER-TO-VPC`: Choose a name for the connection, such as `om-to-vpc`.
   - `OPSMANAGER-NETWORK`: Enter the name of the network assigned to the Ops Manager project in GCP, such as `my-om-project`.
   - `VPC-HOST-PROJECT`: Enter the name you gave the shared VPC project in Step 1: Provision the Shared VPC.
   - `VPC-NETWORK`: Enter the name of the network you gave the shared VPC project in Step 2: Create Shared VPC Networks.

3. Enter the following command, replacing `VPC-HOST-PROJECT` with the new shared VPC project you created in Step 1: Provision the Shared VPC:

   ```
   $ gcloud config set project VPC-HOST-PROJECT
   ```

4. Enter the following command to create a connection from the new shared VPC project to the Ops Manager project:

   ```
   $ gcloud beta compute networks peerings create VPC-TO-OPSMANAGER
     --network VPC-NETWORK
     --peer-project OPSMANAGER-PROJECT
     --peer-network OPSMANAGER-NETWORK
     --auto-create-routes
   ```

   Replace the following text and run the following command:
   - `VPC-TO-OPSMANAGER`: Choose a name for the connection, such as `vpc-to-om`.
   - `VPC-NETWORK`: Enter the name of the network you gave the shared VPC project in Step 2: Create Shared VPC Networks.
   - `OPSMANAGER-PROJECT`: Enter the name of the project that contains your Ops Manager installation.
Step 4: Verify the Shared VPC Configuration

After configuring a shared VPC, use the following procedure to verify that the shared VPC host project VM appears in the Ops Manager project.

1. From https://console.cloud.google.com, select the Ops Manager project from the drop-down menu at the top of the page.

2. Click Networking, then VPC networks.

3. Confirm that the shared VPC network name appears in the Subnets list.

4. Confirm that the shared VPC network IP address ranges match what you set for the new VPC project in Step 2: Create a Shared VPC Network.
Deleting PCF from GCP

Page last updated:

When you deploy Pivotal Cloud Foundry (PCF) to Google Cloud Platform (GCP), you provision a set of resources. This topic describes how to delete the resources associated with a PCF deployment.

You can delete the resources in one of two ways:

- If you created a separate project for your PCF deployment, perform the procedure in the Delete the Project delete the project.
- If the project that contains your PCF deployment also contains other resources that you want to preserve, perform the procedure in the Delete PCF Resources section.

Delete the Project

Perform the following steps to delete the project for your PCF deployment:

1. Navigate to the GCP Console Dashboard.
2. Under your Project, click Manage project settings.
3. Click DELETE PROJECT.
4. Enter your project ID and click SHUT DOWN to confirm.

Delete PCF Resources

Perform the following steps to delete the resources associated with your PCF deployment:

1. Navigate to the GCP Console Dashboard.
2. Click the upper left icon and select Networking.
3. Click Load balancing.
4. Perform the following steps for all load balancers associated with your PCF deployment:
   a. Click the trashcan icon next to the load balancer.
   b. In the next dialog, select any health checks and backend services associated with the load balancer.
   c. Click DELETE LOAD BALANCER AND THE SELECTED RESOURCES.
5. Click the upper left icon and select Compute Engine.
6. Perform the following steps for VM instances, Instance groups, and Disks:
   a. Select the checkbox next to the PCF resource.
   b. When all PCF resources are selected, click DELETE in the upper right.
   c. Click DELETE to confirm.
7. Click the upper left icon and select Networking.
8. Click External IP addresses.
9. Select all external IP addresses associated with your PCF deployment, and click RELEASE STATIC ADDRESS.
10. Click on Networks, and perform the following steps for any networks you created for PCF:
    a. Click the name of the network.
    b. Click DELETE NETWORK.
    c. Click DELETE to confirm.
11. Click the upper left icon and select IAM & Admin.
12. Click the trashcan icon next to the service account you created for PCF and click REMOVE.
13. Navigate to Compute Engine > Metadata > SSH Keys Delete the vcap SSH key that you created for the project.
Troubleshooting PCF on GCP

This topic describes how to troubleshoot known issues when deploying Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

Problems Connecting with Single Sign-On (SSO)

Users may be unable to connect to applications running on PCF using SSO.

Explanation
SSO does not support multi-subnets.

Solution
Ensure that you have configured only one subnet. See Preparing to Deploy Ops Manager on GCP Manually.

Uploading PAS Tile Causes Ops Manager Rails Application Crash

Uploading the Pivotal Application Service (PAS) tile causes the Ops Manager Rails application to crash.

Explanation
In compressed format, the PAS tile is 5 GB in size. However, when uncompressed during installation, the PAS tile requires additional disk space that can exhaust the space allocated to the boot disk.

Solution
Ensure that the boot disk is allocated at least 50 GB of space. See Step 3: Create the Ops Manager VM Instance in Deploying Ops Manager on GCP Manually.

PAS Deployment Fails - MySQL Monitor replication-canary Job

During installation of the PAS tile, the replication-canary job fails to start. The error reported in the installation log resembles the following:

```
Started updating job mysql_monitor > mysql_monitor/0
{4be7ec82-3c0f-41af-9d0f-90d1f12683c8} (canary). Failed: 'mysql_monitor/0
{4be7ec82-3c0f-41af-9d0f-90d1f12683c8}' is not running after update.
Review logs for failed jobs: replication-canary (00:05:13)
```

Explanation
This error can appear as a result of incorrect configuration of network traffic and missed communication between the Gorouter and a load balancer.

Solution
1. Make sure you have selected the **Forward SSL to PAS Router** option in your PAS Network Configuration.

2. Verify that you have configured the firewall rules properly and that TCP ports [80, 443, 2222, and 8080] are accessible on your GCP load balancers. See Step 5: Create Firewall Rules for the Network in Preparing to Deploy Ops Manager on GCP Manually.

3. Verify that you have configured the proper SSL certificates on your HTTP(S) load balancer in GCP. See Step 8: Create HTTP Load Balancer in Preparing to Deploy Ops Manager on GCP Manually.

4. If necessary, re-upload a new certificate and update any required SSL Certificate and SSH Key fields in your PAS Network Configuration.

---

**Insufficient External Database Permissions**

Upgrade issues can be caused when the external database user used for the network policy DB is given insufficient permissions. To avoid this upgrade issue, ensure that the networkpolicyserver database user has the **ALL PRIVILEGES** permission.
Upgrading BOSH Director on GCP

This topic describes how to upgrade BOSH Director for Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP).

In this procedure, you create a new Ops Manager VM instance that hosts the new version of Ops Manager. To upgrade, you export your existing Ops Manager installation into this new VM.

After you complete this procedure, follow the instructions in Upgrading PAS and Other Pivotal Cloud Foundry Products.

Step 1: Locate the Pivotal Ops Manager Installation File

1. Log in to the Pivotal Network, and click on Pivotal Cloud Foundry Operations Manager.

2. From the Releases drop-down, select the release for your upgrade.

3. Select one of the following download files:
   - Pivotal Cloud Foundry Ops Manager for GCP
   - Pivotal Cloud Foundry Ops Manager YAML for GCP When you click on the download link, your browser downloads or opens the OpsManager_version_onGCP.pdf or OpsManager_version_onGCP.yml file.

   These documents provide the GCP location of the Ops Manager.tar.gz installation file based on the geographic location of your installation.

4. Copy the filepath string of the Ops Manager image based on your existing deployment location.

Step 2: Create a Private VM Image

1. Log in to the GCP Console.

2. In the left navigation panel, click Compute Engine, and select Images.

3. Click Create Image.

4. Complete the following fields:
   - Name: Enter a name that matches the naming convention of your existing Ops Manager image files.
   - Encryption: Leave Automatic (recommended) selected.
   - Source: Choose Cloud Storage file.
   - Cloud Storage file: Paste in the Google Cloud Storage filepath you copied from the PDF or YAML file in the previous step.
5. Click **Create**. The file may take a few minutes to import.

**Step 3: Create the Ops Manager VM Instance**

1. Select the checkbox for the image that you created above.

2. Click **Create Instance**.

3. In the **Create an instance form**, complete the following fields:
   - **Name**: Enter a name that matches the naming conventions of your existing deployment.
   - **Zone**: Choose a zone from the region of your existing deployment.
   - **Machine type**: Click **Customize** to manually configure the vCPU and memory. An Ops Manager VM instance requires the following minimum specifications:

<table>
<thead>
<tr>
<th>Machine Spec</th>
<th>Minimum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2 vCPUs</td>
</tr>
<tr>
<td>Memory</td>
<td>8 GB</td>
</tr>
</tbody>
</table>

   - **Boot disk**: Click **Change**, then perform the following steps:
     - Click **Custom images** if it is not already selected.
     - Select the **Boot disk type**. If you have an Ops Manager environment with high performance needs, select **SSD**. As an example, environments used to develop PCF tiles may benefit from a higher performing Ops Manager VM boot disk. For most environments, however, you can select **Standard**.
     - Set the **Size (GB)** of the boot disk to the minimum or higher.

<table>
<thead>
<tr>
<th>Machine Spec</th>
<th>Minimum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot disk</td>
<td>100 GB</td>
</tr>
</tbody>
</table>
Click Select to save.

- Under Identity and API access, choose the Service account you created when you initially installed Pivotal Cloud Foundry. See Step 1: Set up IAM Service Accounts in Preparing to Deploy Ops Manager on GCP Manually.
- Allow HTTP traffic: Only select this checkbox if you selected it in your original Ops Manager VM configuration.
- Allow HTTPS traffic: Only select this checkbox if you selected it in your original Ops Manager VM configuration.

Networking: Select the Networking tab, and perform the following steps:

- For Network and Subnetwork, select the network and subnetwork you created when you initially deployed Pivotal Cloud Foundry. See Step 3: Create a GCP Network with Subnets in Preparing to Deploy Ops Manager on GCP Manually.
- For Network tags, enter any tags that you applied to your original Ops Manager. For example, if you used the `pcf-opsmanager` tag to apply the firewall rule you created in Step 5: Create Firewall Rules for the Network, then apply the same tag to this Ops Manager VM.
- For Internal IP, select Custom. In the Internal IP address field, enter a spare address located within the reserved IP range you configured in your existing BOSH Director as part of Step 5: Create Networks Page. Do not use 10.0.0.1, which is configured for the...
Gateway,

- For External IP, select New static IP address.... In the next form, enter a name for the static IP. For example, om-public-ip. Click Reserve. In the External IP drop-down, select the static IP address you just reserved.

4. Click Create to deploy the new Ops Manager VM. This may take a few moments.

5. Navigate to your DNS provider, and modify the entry that points a fully qualified domain name (FQDN) to the Ops Manager VM. Replace the original Ops Manager static IP address with the public IP address of the new Ops Manager VM you created in a previous step.

⚠️ warning: In order to set up Ops Manager authentication correctly, Pivotal recommends using a Fully Qualified Domain Name (FQDN) to access Ops Manager. Using an ephemeral IP address to access Ops Manager can cause authentication errors upon subsequent access. If you are importing older Ops Manager settings, you must use the FQDN.

Next Steps

After you complete this procedure, continue the upgrade instructions in Upgrading Pivotal Cloud Foundry topic.

Later on, if you need to SSH into the Ops Manager VM to perform diagnostic troubleshooting, see SSH into Ops Manager.
Installing Pivotal Cloud Foundry on OpenStack

Overview

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter. For guidance on OpenStack service credential management, see Open Stack Security Documents below.

OpenStack Security Documents

These documents provide a general reference for OpenStack service credential management.

- OpenStack credential configuration
- OpenStack credential creation
- OpenStack deployment configuration

Requirements

This section describes the requirements for installing PCF on OpenStack, including general requirements for installing PCF with Ops Manager and Pivotal Application Service (PAS) as well as OpenStack requirements.

Note: You can install PCF on OpenStack with the Pivotal Application Service (PAS) runtime. The Pivotal Container Service (PKS) runtime is not supported for OpenStack. For more information about PAS, see PAS Concepts. For more information about PKS, see Pivotal Container Service (PKS).

General Resource Requirements

The following are general requirements for deploying and managing a PCF deployment with Ops Manager and Pivotal Application Service (PAS):

- A wildcard DNS record that points to your router or load balancer. Alternatively, you can use a service such as xip.io. For example, 203.0.113.0.xip.io.
  - PAS gives each application its own hostname in your app domain.
  - With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record *.example.com pointing to your load balancer or router, every application deployed to the example.com domain resolves to the IP address of your router.
  - At least one wildcard TLS certificate that matches the DNS record you set up above, *.example.com.

- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your gorouters
  - One static IP address for each job in the Ops Manager tile. See the Resource Config pane for each tile for a full list.
  - One static IP address for each job listed below:
    - Consul
    - NATS
    - File Storage
    - MySQL Proxy
    - MySQL Server
    - Backup Restore Node
    - HAProxy
    - Router
    - MySQL Monitor
    - Diego Brain
    - TCP Router
One IP for each VM instance created by the service.

An additional IP address for each compilation worker. So the formula for total IPs needed is

\[
\text{IPs needed} = \text{static IPs} + \text{VM instances} + \text{compilation workers}
\]

Note: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

One or more NTP servers if not already provided by your IaaS.

(Recommended) A network without DHCP available for deploying the PAS VMs.

Note: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

(Optional) External storage. When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS tile. Pivotal recommends using external storage whenever possible. See Configure File Storage for a discussion of how file storage location affects platform performance and stability during upgrades.

(Optional) External databases. When you deploy PCF, you can select internal or external databases for the BOSH Director and for PAS. Pivotal recommends using external databases in production deployments. An external database must be configured to use the UTC timezone.

(Optional) External user stores. When you deploy PCF, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for PAS, to integrate existing user accounts.

The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

OpenStack Requirements

The following are OpenStack requirements for deploying PCF:

- PCF is supported on the OpenStack Liberty, Mitaka, and Newton releases. OpenStack is a collection of inter-operable components and requires general OpenStack expertise to troubleshoot issues that may occur when installing Pivotal Cloud Foundry on particular releases and distributions. To verify that your OpenStack platform is compatible with PCF, use the OpenStack Validator tool. To access the OpenStack Validator tool, see CF OpenStack Validator on GitHub.

- Pivotal recommends granting complete access to the OpenStack logs to the operator managing the PCF installation process.

- For OpenStack accounts for PCF, Pivotal recommends following the principle of least privilege by scoping privileges to the most restrictive permissions possible for a given role.

- You must have a dedicated OpenStack project, formerly known as an OpenStack tenant.

- You must have Keystone access to the dedicated OpenStack project, including the following:
  
  - Auth URL
  - Username and password. The PrimaryProject for the user must be the project you want to use to deploy PCF. For more information, see Manage projects and users in the OpenStack documentation.
  - Project name
  - Region (with multiple availability zones if you require high availability)
  - SSL certificate for your wildcard domain (see below)

- You must have the ability to do the following in OpenStack:
  
  - Create and modify VM flavors
  - Enable DHCP if required
  - Create a network and then connect that network with a router to an external network
  - Create an external network with a pool of floating IP addresses
  - Boot VMs directly from image
  - Create two wildcard domains for separate system and app domains

- The following are resource requirements for the dedicated OpenStack project:
  
  - 118 GB of RAM
  - 22 available instances
  - 16 small VMs (1 vCPU, 1024 MB of RAM, 10 GB of root disk)
  - 3 large VMs (4 vCPU, 16384 MB of RAM, 10 GB of root disk)
  - 3 extra-large VMs (8 vCPU, 16 GB of RAM, 160 GB of ephemeral disk)
  - 56 vCPUs
  - 1 TB of storage

Note: Pivotal recommends that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install PAS, and connect to services.

Note: If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

© Copyright Pivotal Software Inc, 2013-2019
Nova or Neutron networking with floating IP support

The following are requirements for the OpenStack Cinder back-end:

- PCF requires RAW root disk images. The Cinder back-end for your OpenStack project must support RAW.
- Pivotal recommends that you use a Cinder back-end that supports snapshots. This is required for some BOSH functionalities.
- Pivotal recommends enabling your Cinder back-end to delete block storage asynchronously. If this is not possible, it must be able to delete multiple 20 GB volumes within 300 seconds.

The following are requirements for using an Overlay Network with VXLAN or GRE Protocols:

- If an overlay network is being used with VXLAN or GRE protocols, the MTU of the created VMs must be adjusted to the best practices recommended by the plugin vendor (if any).
- DHCP must be enabled in the internal network for the MTU to be assigned to the VMs automatically.
- Review the Configuring PAS topic to adjust your MTU values.
- Failure to configure your overlay network correctly could cause Apps Manager to fail since applications will not be able to connect to the UAA.

Install PCF on OpenStack with PAS

To install PCF on OpenStack with the PAS runtime, do the following:

1. Deploy Ops Manager. See Deploying Ops Manager on OpenStack.
2. Configure BOSH Director on OpenStack. See Configuring BOSH Director on OpenStack.
3. Configure PAS. See Configuring PAS.

Note: By default, PAS deploys the number of VM instances required to run a highly available configuration of PCF. If you are deploying a test or sandbox PCF that does not require HA, then you can scale down the number of instances in your deployment. For information about the number of instances required to run a minimal, non-HA PCF deployment, see Scaling PAS.

Note: If you are using IPsec, your resource usage will increase by approximately 36 bytes. View the Installing IPsec topic for information, including setting correct MTU values.
Deploying Ops Manager to OpenStack

This guide describes how to provision the OpenStack infrastructure where you need to install Pivotal Cloud Foundry. Use this topic when Installing Pivotal Cloud Foundry on OpenStack.

After completing this procedure, complete all of the steps in the Configuring BOSH Director on OpenStack and Configuring PAS topics.

Note: This document uses Mirantis OpenStack for screenshots and examples. The screens of your OpenStack vendor configuration interface may differ.

Step 1: Log in to the OpenStack Horizon Dashboard

1. Log in to the OpenStack Horizon dashboard.

2. Click Connect.

3. From the OpenStack project list dropdown, set the active project by selecting the project where you will deploy PCF.

Step 2: Configure Security

⚠️ warning: If you are using OpenStack Liberty or Mitaka, do not create the key pair with the OpenStack Horizon dashboard. Instead make sure that you generate the SSH key pair manually. For example, use the ssh-keygen command. Then follow the procedure below to import that key pair into OpenStack. This is due to an OpenStack bug.

© Copyright Pivotal Software Inc, 2013-2019
1. In the left navigation of your OpenStack Horizon dashboard, click Project > Compute > Access & Security.

2. Select the Key Pairs tab on the Access & Security page.

3. Click Import Key Pair.

4. Enter a Key Pair Name and the contents of your public key in the Public Key field.

5. Click Import Key Pair.

6. In the left navigation, click Access & Security to refresh the page. The new key pair appears in the list.

7. Select the Security Groups tab. Click Create Security Group and create a group with the following properties:
   - Name: opsmanager
   - Description: Ops Manager

8. Select the checkbox for the opsmanager Security Group and click Manage Rules.
9. Add the following ingress access rules for HTTP, HTTPS, and SSH as shown in the table below. The rules with opsmanager in the Remote column have restricted access to that particular Security Group.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Ether Type</th>
<th>IP Protocol</th>
<th>Port/Port Range</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>22 (SSH)</td>
<td>0.0.0.0/0 (CIDR)</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>80 (HTTP)</td>
<td>0.0.0.0/0 (CIDR)</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>443 (HTTPS)</td>
<td>0.0.0.0/0 (CIDR)</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>4222 (NATS)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>6868 (BOSH Agent)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>8844 (CredHub)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>8853 (BOSH Health Monitor)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>25250 (BOSH Blobstore)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>25555 (BOSH Director)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Ingress</td>
<td>IPv4</td>
<td>TCP</td>
<td>25777 (BOSH Registry)</td>
<td>opsmanager</td>
</tr>
<tr>
<td>Egress</td>
<td>IPv4</td>
<td>TCP</td>
<td>1-65535</td>
<td>0.0.0.0/0 (CIDR)</td>
</tr>
</tbody>
</table>

Note: Adjust the remote sources as necessary for your own security compliance. Pivotal recommends limiting remote access to Ops Manager to IP ranges within your organization.

10. Leave the existing default egress access rules as shown in the screenshot below.
Step 3: (Optional) Run the CF OpenStack Validator Tool

As an optional but recommended step, you can now run the CF OpenStack Validator tool against your OpenStack tenant to verify support for PCF.

1. Follow the directions for running the [CF OpenStack Validator Tool](#).
2. When configuring the CPI version used by the Validator, specify the OpenStack CPI version indicated in the [PCF Ops Manager Release Notes](#) for the PCF release that you are planning to deploy.

Troubleshooting the output of the CF OpenStack Validator tool is beyond the scope of this document.

---

Step 4: Create Ops Manager Image

You can create the Ops Manager image in OpenStack using the OpenStack Horizon dashboard.

Note: If your Horizon Dashboard does not support file uploads, you must use the [Glance CLI](#) client.

To create an Ops Manager image in OpenStack, perform the following steps:

1. Download the [Pivotal Cloud Foundry Ops Manager for OpenStack](#) image file from [Pivotal Network](#).
2. In the left navigation of your OpenStack dashboard, click *Project > Compute > Images*.
3. Click Create Image. Complete the Create An Image page with the following information:

   - **Name**: Enter *Ops Manager*.
   - **Image Source**: Select *Image File*.
   - **Image File**: Click Choose File. Browse to and select the image file that you downloaded from [Pivotal Network](#).
   - **Format**: Select *Raw*.
   - **Minimum Disk (GB)**: Enter *80*.
   - **Minimum RAM (MB)**: Enter *8192*.
   - Deselect the *Public* checkbox.
   - Select the *Protected* checkbox.
4. Click Create Image.

Step 5: Launch Ops Manager VM

1. In the left navigation of your OpenStack dashboard, click Project > Compute > Images.

2. Click Launch.

3. In the Details tab, specify the following values:
   - **Instance Name**: Enter *Ops Manager*.
   - **Availability Zone**: Use the dropdown to select an availability zone. You specify this availability zone in the Complete the Availability Zones Pages step of Configuring Ops Manager Director.
   - **Count**: Do not change from the default value of 1.
4. In the **Source** tab, specify the following values:
   - **Select Boot Source**: Select **Image**.
   - **Create New Volume**: Leave **No** selected.
   - **Allocated**: Make sure **Ops Manager** is selected.

5. In the **Flavor** tab, configure the OpenStack VM flavors as follows:

   **Note**: Do not change the names of the VM flavors.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Memory_MB</th>
<th>Disk</th>
<th>Ephemeral</th>
<th>VCPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m1.small</td>
<td>2048</td>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
6. In the Networks tab, select a private subnet. You add a Floating IP to this network in a later step.

7. Skip the Network Ports tab.


9. In the Key Pair tab, select the key pair that you imported in Step 2: Configure Security.
10. Skip the Configuration and Metadata tabs.

11. Click Launch Instance. This step starts your new Ops Manager instance.

**Step 6: Associate a Floating IP Address**

1. In the left navigation of your OpenStack dashboard, click Project > Compute > Instances.

2. Wait until the Power State of the Ops Manager instance shows as Running.

3. Record the private IP Address of the Ops Manager instance.

You must provide this IP Address when you perform Step 6: Complete the Create Networks Page in Ops Manager.

4. Select the Ops Manager checkbox. Click the Actions dropdown and select Associate Floating IP. The Manage Floating IP Associations screen
5. Under **IP Address**, click the plus button (+). The **Allocate Floating IP** screen appears.

6. Under **Pool**, select an IP Pool and click **Allocate IP**.

7. Under **Port to be associated**, select your **Ops Manager** instance.

8. Click **Associate**.

**Step 7: Add Blob Storage**

1. In the left navigation of your OpenStack dashboard, click **Project > Object Store > Containers**.

2. Click **Create Container**. Create a container with the following properties:
   - **Container Name**: Enter *pcf*.
   - **Container Access**: Leave *public* unselected.
3. Click Create.

Step 8: Download Credentials for S3 Blob Storage

1. In the left navigation of your OpenStack dashboard, click Project > Compute > Access & Security. Select the API Access tab.

2. Click Download EC2 Credentials.

3. Unzip the downloaded credentials.

4. If you select S3 Compatible Blobstore in your BOSH Director Config, you need the contents of this file to complete the configuration.

Step 9: Create a DNS Entry

Note: For security, Ops Manager v1.7 and later require you to create a fully qualified domain name in order to access Ops Manager during the initial configuration.

Create a DNS entry for the floating IP address that you assigned to Ops Manager in Step 6: Associate a Floating IP Address.

You must use this fully qualified domain name when you log into Ops Manager for the first time.

Step 10: Configure BOSH Director for OpenStack

After completing this procedure, complete all of the steps in the Configuring BOSH Director on OpenStack and Configuring PAS topics.

Return to Installing Pivotal Cloud Foundry on OpenStack.
Configuring BOSH Director on OpenStack

This topic describes how to configure BOSH Director after deploying Pivotal Cloud Foundry (PCF) on OpenStack. Use this topic when Installing Pivotal Cloud Foundry on OpenStack.

- Before beginning this procedure, ensure that you have successfully completed all steps in Provisioning the OpenStack Infrastructure.
- After you complete this procedure, follow the instructions in Configuring PAS.

Note: You can also perform the procedures in this topic using the Ops Manager API. For more information, see Using the Ops Manager API.

Step 1: Access Ops Manager

1. In a web browser, navigate to the fully qualified domain you created in the Create a DNS Entry step of Provisioning the OpenStack Infrastructure.

2. When Ops Manager starts for the first time, you must choose one of the following:

   - **Internal Authentication**: If you use Internal Authentication, Ops Manager maintains your user database.
   - **SAML Identity Provider**: If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
   - **LDAP Server**: If you use a LDAP Server, an external identity server maintains your user database.

### Internal Authentication

1. When redirected to the Internal Authentication page, do the following:
   - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
   - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   - Read the **End User License Agreement**, and select the checkbox to accept the terms.
   - Click **Setup Authentication**.
2. Log in to Ops Manager with the Admin username and password you created in the previous step.

SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.
3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **6a.** Ops Manager SAML service provider metadata: [https://OPS-MAN-FQDN:443/uaa/saml/metadata](https://OPS-MAN-FQDN:443/uaa/saml/metadata)
   - **6b.** BOSH Director SAML service provider metadata: [https://BOSH-IP-ADDRESS:8443/saml/metadata](https://BOSH-IP-ADDRESS:8443/saml/metadata)

   **Note:** To retrieve your **BOSH-IP-ADDRESS**, navigate to the **Status** tab in the **BOSH Director** tile. Record the **BOSH Director** IP address.

7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.
   - **Audience URI (SP Entity ID)**: [https://OP-MAN-FQDN:443/uaa](https://OP-MAN-FQDN:443/uaa)
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.
   - **Single sign on URL**: [https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP](https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP)
   - **Audience URI (SP Entity ID)**: [https://BOSH-IP:8443](https://BOSH-IP:8443)
   - **Name ID**: Email Address
   - **SAML authentication requests are always signed**

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

### LDAP Server

- **For Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - **ldap://**: This specifies that the LDAP server uses an unencrypted connection.
  - **ldaps://**: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or
that you import a trusted certificate to the JVM truststore.

- For **LDAP Username** and **LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: cn=administrator,ou=Users,dc=example,dc=com

  Note: Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

  warning: Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: ou=Users,dc=example,dc=com

- For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter \( \text{cn} = \text{Smith} \) returns all objects with a common name equal to \text{Smith}.

  In the LDAP search filter string that you use to configure your runtime, use \text{cn}=[0] instead of the username. For example, use \text{cn}=[0] to return all LDAP objects with the same common name as the username.

  In addition to \text{cn} , other attributes commonly searched for and returned are \text{mail} , \text{uid} , and, in the case of Active Directory, \text{sAMAccountName}.

  Note: For instructions for testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry in the Pivotal Support Knowledge Base.

- For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

  For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: \ou=Groups,dc=example,dc=com

  - For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is \text{member}=[0].
  - For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example \text{mail}.
  - For **LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.
  - From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
    - Automatically follow any referrals.
    - Ignore referrals and return partial result.
    - Throw exception for each referral and abort.

  - For **Server SSL Cert** , paste in the root certificate from your CA certificate or your self-signed certificate.

  - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

  - If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

  - Read the **End User License Agreement**, and select the checkbox to accept the terms.

  - Select **Provision an admin client in the BOSH UAA**. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

  - Click **Setup Authentication**.

  - Return to the BOSH Director tile, and continue with the configuration steps below.
Step 2: OpenStack Configs Page

1. In the left navigation of your OpenStack dashboard, click Project > Compute > Access & Security. Select the API Access tab.

2. Record the Service Endpoint for the Identity service. You use this Service Endpoint as the Authentication URL for Ops Manager in a later step.

3. In the PCF Ops Manager Installation Dashboard, click the BOSH Director tile.

4. Select OpenStack Configs.

5. Complete the OpenStack Management Console Config page with the following information:
   - Name: Enter a unique name for the OpenStack config.
   - Authentication URL: Enter the Service Endpoint for the Identity service that you recorded in a previous step.
   - Keystone Version: Choose a Keystone version, either v2 or v3.
     - If you choose v3, enter the OpenStack Keystone domain to authenticate against in the Domain field. For more information about Keystone domains in OpenStack, see Domains in the OpenStack documentation.
Username: Enter your OpenStack Horizon username. The PrimaryProject for the user must be the project you are using to deploy PCF. For more information, see Manage projects and users in the OpenStack documentation.

Password: Enter your OpenStack Horizon password.

Tenant: Enter your OpenStack tenant name.

Region: Enter RegionOne, or another region if recommended by your OpenStack administrator.

Select OpenStack Network Type: Select either Nova, the legacy OpenStack networking model, or Neutron, the newer networking model.

Ignore Server Availability Zone: Do not select the checkbox.

Security Group Name: Enter opsmanager. You created this Security Group in the Configure Security step of Provisioning the OpenStack Infrastructure.

Key Pair Name: Enter the name of the key pair that you created in the Configure Security step of Provisioning the OpenStack Infrastructure.
Step 3: (Optional) Advanced Config Page

1. In Ops Manager, select Advanced Infrastructure Config.

2. If your OpenStack environment requires specific connection options, enter them in the Connection Options field in JSON format. For example:

   ```json
   "connection_options" => { "read_timeout" => 200 }
   ```
3. Click **Save**.

**Step 4: Director Config Page**

1. In Ops Manager, select **Director Config**.

2. Enter one or more NTP servers in the **NTP Servers (comma delimited)** field. For example, `us.pool.ntp.org`.

   **Note:** The NTP server configuration only updates after VM recreation. Ensure that you select the **Recreate all VMs** checkbox if you modify the value of this field.

3. Leave the **JMX Provider IP Address** field blank.

4. Leave the **Bosh HM Forwarder IP Address** field blank.

   **Note:** Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the **Bosh HM Forwarder IP Address** field blank.

5. Select the **Enable VM Resurrector Plugin** checkbox to enable the BOSH Resurrector functionality and increase your runtime availability.

6. Select **Enable Post Deploy Scripts** to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   **Note:** If you intend to install, you must enable post-deploy scripts.

7. Select **Recreate all VMs** to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select **Recreate All Persistent Disks** to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.
9. Select **Enable bosh deploy retries** to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select **Skip Director Drain Lifecycle** to prevent drain scripts from running when the BOSH Director is recreated.

11. Select **Store BOSH Job Credentials on tmpfs (beta)** to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. (Optional) Disable **Allow Legacy Agents** if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS secure communications.

13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


   ![HM Pager Duty Plugin](image)

   - **Service Key**: Enter your API service key from PagerDuty.
   - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

   ![HM Email Plugin](image)

   - **Host**: Enter your email hostname.
   - **Port**: Enter your email port number.
   - **Domain**: Enter your domain.
   - **From**: Enter the address for the sender.
   - **Recipients**: Enter comma-separated addresses of intended recipients.
   - **Username**: Enter the username for your email server.
   - **Password**: Enter the password for your email server.
- **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in [Preparing CredHub HSMs for Configuration](#).

**Note:** After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.

- **Internal**: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.
- **Luna HSM**: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

  1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
  2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause…

© Copyright Pivotal Software Inc, 2013-2019 356 2.5
service downtime. For this value and the ones below, use values gathered in Preparing CredHub HSMs for Configuration.

3. Provider Partition Password
4. Provider Client Certificate: The certificate that validates the identity of the HSM when CredHub connects as a client.
5. Provider Client Certificate Private Key
6. HSM Host Address
7. HSM Port Address: If you do not know your port address, enter 1792.
8. Partition Serial Number
9. HSM Certificate: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. Select a Blobstore Location to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable and less secure, Pivotal recommends that you configure an external blobstore.

Note: After you deploy Ops Manager, you cannot change the blobstore location.

- Internal: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.
- Enable TLS: Select this checkbox to enable TLS to the blobstore.

Note: If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.
**S3 Compatible Blobstore**: Select this option to use an external S3-compatible endpoint. Follow the procedures in [Sign up for Amazon S3](#) and [Creating a Bucket](#) in the AWS documentation. When you have created an S3 bucket, complete the following steps:

1. **S3 Endpoint**: Navigate to the [Regions and Endpoints](#) topic in the AWS documentation.
   a. Locate the endpoint for your region in the [Amazon Simple Storage Service (S3)](#) table and construct a URL using your region's endpoint. For example, if you are using the us-west-2 region, the URL you create would be `https://s3-us-west-2.amazonaws.com`. Enter this URL into the `S3 Endpoint` field.
   b. On a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.
   c. Copy the custom public CA certificate you used to sign the S3 endpoint into `/etc/ssl/certs` on the Ops Manager VM.
   d. On the Ops Manager VM, run `sudo update-ca-certificates -f -v` to import the custom CA certificate into the Ops Manager VM truststore.

   **Note**: You must also add this custom CA certificate into the `Trusted Certificates` field in the `Security` page. See [Security Page](#) for instructions.

2. **Bucket Name**: Enter the name of the S3 bucket.
3. **Access Key** and **Secret Key**: Enter the keys you generated when creating your S3 bucket.
4. **Select V2 Signature** or **V4 Signature**. If you select V4 Signature, enter your **Region**.

   **Note**: AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see [Authenticating Requests](#) in the AWS documentation.

**GCS Blobstore**: Select this option to use an external GCS endpoint. To create a GCS bucket, you must have a GCS account. Follow the procedures in [Creating Storage Buckets](#) in the GCS documentation to create a GCS bucket. When you have created a GCS bucket, complete the following steps:

1. **Bucket Name**: Enter the name of your GCS bucket.
2. **Storage Class**: Select the storage class for your GCS bucket. See [Storage Classes](#) in the GCP documentation for more information.
3. **Service Account Key**: Follow the steps in the [Set up IAM Service Accounts](#) section of [Preparing to Deploy Ops Manager on GCP Manually](#) to download a JSON file with a private key. Enter the contents of the JSON file into the field.

18. Select a **Database Location**. By default, Ops Manager deploys and manages an **Internal** database for you. If you choose to use an **External MySQL Database**, complete the associated fields with information obtained from your external MySQL Database provider: **Host**, **Port**, **Username**, **Password**, and **Database**. In addition, if you selected the **Enable TLS for Director Database** checkbox, you can complete the following optional fields:

- **Enable TLS**: Select this checkbox enables TLS communication between the BOSH Director and the database.
- **TLS CA**: Enter the Certificate Authority for the TLS Certificate.
- **TLS Certificate**: Enter the client certificate for mutual TLS connections to the database.
- **TLS Private Key**: Enter the client private key for mutual TLS connections to the database.
- **Advanced DB Connection Options**: If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.

19. (Optional) Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to 5.

20. (Optional) **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. (Optional) To add a custom URL for your BOSH Director, enter a valid hostname in **Director Hostname**. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director in the Pivotal Support Knowledge Base.

![Director Configuration Options](image)

22. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the **Custom SSH Banner** field.

23. (Optional) Enter your comma-separated custom **Identification Tags**. For example, `iaas:foundation1, hello:world`. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

24. Click **Save**.

**Step 5: Create Availability Zones Page**

1. In Ops Manager, select **Create Availability Zones**.

![Availability Zones](image)

2. Enter the name of the availability zone that you selected in the **Launch Ops Manager VM** step of Provisioning the OpenStack Infrastructure.

3. (Optional) Select an OpenStack config name from the **IaaS Configuration** dropdown. The default is set to your first OpenStack config.
4. Enter the **OpenStack Availability Zone** of your OpenStack environment. Many OpenStack environments default to `nova`.

5. Click **Add** for each additional OpenStack config you created in **Step 2: OpenStack Configs Page**. Give each AZ a unique **Name** and an **IaaS Configuration** with a different OpenStack config.

6. Click **Save**.

**Step 6: Create Networks Page**

1. In the left navigation of your OpenStack dashboard, click **Project > Network > Networks**.

2. Click the name of the network that contains the private subnet where you deployed the Ops Manager VM. The OpenStack Network Detail page displays your network settings.

3. In Ops Manager, select **Create Networks**.
4. Select Enable ICMP checks to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable. Review the Configure Security step of Deploying BOSH and Ops Manager to OpenStack to ensure you have configured ICMP in your Security Group.

5. Use the following steps to create one or more Ops Manager networks using information from your OpenStack network:

- Click Add Network.
- Enter a unique Name for the network.
- Click Add Subnet to create one or more subnets for the network.
- For Network ID, use the ID from the OpenStack page.
- For CIDR, use the Network Address from the OpenStack page.
- For Reserved IP Ranges, use the first 10 IP addresses of the Network Address range, and the private IP address of the Ops Manager instance that you recorded in the Associate a Floating IP Address step of Provisioning the OpenStack Infrastructure.
- For DNS, enter one or more Domain Name Servers.
- For Gateway, use the Gateway IP from the OpenStack page.
- For Availability Zones, select which Availability Zones to use with the network.
6. Click Save.

Note: After you deploy Ops Manager, you add subnets with overlapping Availability Zones to expand your network. For more information about configuring additional subnets, see Expanding Your Network with Additional Subnets.

Step 7: Assign AZs and Networks Page

1. Select Assign Availability Zones.

![Assign AZs and Networks](image)

2. From the Singleton Availability Zone dropdown, select the availability zone that you created in a previous step. The BOSH Director installs in this Availability Zone.

3. Use the dropdown to select the Network that you created in a previous step. BOSH Director installs in this network.

4. Click Save.

Step 8: Security Page

2. In Trusted Certificates, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEF12345678ABCDEF12345678ABCDEF12345678ABCDEF12345678
ABCD12345678ABCD12345678ABCD12345678ABCD12345678...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
BCDEF12345678BCDEF12345678BCDEF12345678BCDEF12345678
BCDE12345678BCDE12345678BCDE12345678BCDE12345678...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
CDEFG12345678CDEFG12345678CDEFG12345678CDEFG12345678
CDEF12345678CDEF12345678CDEF12345678CDEF12345678...
-----END CERTIFICATE-----
```

3. (Optional) Select the Include OpsManager Root CA in Trusted Certs checkbox to include the Ops Manager root CA in the Trusted Certificates field. BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose Generate passwords or Use default BOSH password. Pivotal recommends that you use the Generate passwords option for greater security.

5. Click Save. To view your saved Director password, click the Credentials tab.

Step 9: BOSH DNS Config Page
1. Select **BOSH DNS Config**.

2. (Optional) In **Excluded Recursors**, enter a list of prohibited recursor addresses.

3. (Optional) In **Recursor Timeout**, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   **Note**: This time limit must include one of the Go parse duration time units. For example, entering `5s` sets the timeout limit to five seconds. For more information about supported time units, see `func ParseDuration` in the Go Programming Language documentation.

4. (Optional) In **Handlers**, enter a list of custom domain handlers in JSON format.

5. Click **Save**.

---

**Step 10: Syslog Page**
1. Select **Syslog**.

2. (Optional) Select **Yes** to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.
7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** field. This field requires the rainerescript syntax.

10. Click **Save Syslog Settings**.

**Step 11: Resource Config Page**

1. Select **Resource Config**.

<table>
<thead>
<tr>
<th>JOB</th>
<th>INSTANCES</th>
<th>PERSISTENT DISK TYPE</th>
<th>VM TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager Director</td>
<td>Automatic: 1</td>
<td>Automatic: 50 GB</td>
<td>Automatic: medium disk (cpu: 2, ram: 4 GB)</td>
</tr>
<tr>
<td>Master Compilation Job</td>
<td>Automatic: 4</td>
<td>None</td>
<td>Automatic: large cpu (cpu: 4, ram: 4 GB, dl)</td>
</tr>
</tbody>
</table>

2. Adjust any values as necessary for your deployment, such as increasing the persistent disk size. Select **Automatic** from the dropdown to provision the amount of persistent disk predefined by the job. If the persistent disk field reads **None**, the job does not require persistent disk space.

**Note:** Ops Manager requires a Director VM with at least 8 GB memory.

**Note:** If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

3. Click **Save**.

**Step 12: (Optional) Add Custom VM Extensions**

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see **Managing Custom VM Extensions**.

**Step 13: Complete BOSH Director Installation**

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Review Pending Changes**, then **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.
3. BOSH Director installs. The image shows the **Changes Applied** message that Ops Manager displays when the installation process successfully completes.

4. After you complete this procedure, follow the instructions in [Configuring PAS](#).

Return to [Installing Pivotal Cloud Foundry on OpenStack](#).
Installing Pivotal Cloud Foundry on vSphere

This guide describes how to install Pivotal Cloud Foundry (PCF) on vSphere.

If you experience a problem while following the steps below, refer to the Known Issues topics or to Diagnosing Problems in PCF.

Requirements

This section describes the resource requirements for installing PCF.

Note: If you are using the Cisco Nexus 1000v Switch, refer to the Using the Cisco Nexus 1000v Switch with Ops Manager topic for more information.

Note: When installing Ops Manager on a vSphere environment with multiple ESXi hosts, you must use network-attached or shared storage devices. Local storage devices do not support sharing across multiple ESXi hosts.

Resource Requirements

You can install PCF with one of two runtimes. For more information about the requirements for each runtime, see the following:

- **Pivotal Application Service (PAS)**: PAS on vSphere Requirements. For information requirements when deploying with NSX-T, see the Requirements section of Deploying PAS with NSX-T Networking.

- **Pivotal Container Service (PKS):**
  - Without NSX-T: vSphere Prerequisites and Resource Requirements
  - With NSX-T: See the Preparing to Install PKS on vSphere with NSX-T section of Installing PKS on vSphere with NSX-T Data Center.

vSphere Service Account Requirements

Ops Manager requires read/write permissions to the data center level of the vSphere Inventory Hierarchy to successfully install. Pivotal recommends defining a custom role for the service account that has all privileges for all objects in the data center, including propagating privileges to children.

Since Ops Manager passes all required credentials through to BOSH, you only need one service account with the required vSphere privileges to complete the installation. Setting up separate service accounts for Ops Manager and BOSH is not necessary or recommended.

For more information about service account requirements, see the following topics:

- **PAS**: vSphere Service Account Requirements
- **PKS**: Step 3: Create the BOSH/Ops Manager Service Account of Preparing vSphere Before Deploying PKS

Note: You can also apply the default VMware Administrator System Role to the service account to achieve the appropriate permission level.

vSphere Security Documents

- **vSphere Security guide (PDF)**
  This guide contains best practices for securing and managing a vSphere installation.

Procedure

This section describes the steps for installing PCF.
Step 1: Install Ops Manager

Complete the following procedures to install Ops Manager on vSphere:

1. Deploying Ops Manager to vSphere
2. Configuring BOSH Director on vSphere

Step 2: Install a Runtime

Follow the procedure that corresponds to your desired runtime and networking configuration:

- PAS:
  - Without NSX-T: Configuring PAS
  - With NSX-T: Deploying PAS with NSX-T Networking

  **Note:** You must install the NSX-T tile before you install or Upgrade the PAS tile.

- PKS:
  - Without NSX-T: Installing PKS
  - With NSX-T: Deploying NSX-T for PKS

Additional Configuration

See the following topics for additional configuration options:

- Provisioning a Virtual Disk in vSphere
- Using BOSH Resurrector and vSphere HA
- Availability Zones in vSphere
- Updating NSX Security Group and Load Balancer Information
vSphere Service Account Requirements

This topic describes the minimum privileges required by the vSphere BOSH CPI. A vSphere admin must grant the following privileges to the vSphere service account that Pivotal Cloud Foundry (PCF) uses to manage vSphere resources.

The PCF account needs privileges at both the vCenter server level and the Datacenter level. See Hierarchical Inheritance of Permissions in the VMware documentation for how permission levels and inheritance work in vSphere.

vCenter-Level Privileges

Ops Manager assigns custom attributes to the virtual machines (VMs) it deploys to identify BOSH releases and job index information about each VM. vCenter APIs require vCenter server level access to manage these custom attributes.

The following table summarizes the privileges that a PCF account requires at the vCenter Server instance level. Some of these privileges are inherited, and others must be granted by a vCenter admin:

<table>
<thead>
<tr>
<th>Object</th>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role</td>
<td>Read-only</td>
<td>System.Anonymous</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System.Read</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System.View</td>
</tr>
<tr>
<td>Global</td>
<td>Manage custom attributes</td>
<td>Global.ManageCustomFields</td>
</tr>
<tr>
<td></td>
<td>Register Extensions</td>
<td>Extension.Register</td>
</tr>
<tr>
<td>Profile-Driven Storage</td>
<td>Profile-driven Storage</td>
<td>StorageProfile.Update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StorageProfile.View</td>
</tr>
</tbody>
</table>

Datacenter-Level Privileges

The following privileges must be set at the data center level:

<table>
<thead>
<tr>
<th>Object</th>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore</td>
<td>Low level file operations</td>
<td>Datastore.FileManagement</td>
</tr>
<tr>
<td>Network</td>
<td>Assign network</td>
<td>Network.Assign</td>
</tr>
</tbody>
</table>

Folder and Datastore-Level Privileges

You must grant the following privileges on any entities in a datacenter where you will deploy PCF:

<table>
<thead>
<tr>
<th>Datastore Object</th>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocate space</td>
<td></td>
<td>Datastore.AllocateSpace</td>
</tr>
<tr>
<td>Browse datastore</td>
<td></td>
<td>Datastore.Browse</td>
</tr>
<tr>
<td>Remove file</td>
<td></td>
<td>Datastore.DeleteFile</td>
</tr>
<tr>
<td>Update file</td>
<td></td>
<td>Datastore.UpdateVirtualMachineFiles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Folder Object</th>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Inventory Service Object

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>vSphere Tagging &gt; Create vSphere Tag</td>
<td>InventoryService.Tagging.CreateTag</td>
</tr>
<tr>
<td>vSphere Tagging &gt; Delete vSphere Tag</td>
<td>InventoryService.Tagging.EditTag</td>
</tr>
<tr>
<td>vSphere Tagging &gt; Edit vSphere Tag</td>
<td>InventoryService.Tagging.DeleteTag</td>
</tr>
</tbody>
</table>

## Resource Object

When using `vAppImport` to clone a VM, BOSH requires the resource migration privileges to create a new, powered-off VM based on a given stemcell. BOSH migrates the VM to the destination datastore, where Ops Manager deploys the VM and powers it on.

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign virtual machine to resource pool</td>
<td>Resource.AssignVMToPool</td>
</tr>
<tr>
<td>Migrate powered off virtual machine</td>
<td>Resource.ColdMigrate</td>
</tr>
<tr>
<td>Migrate powered on virtual machine</td>
<td>Resource.HotMigrate</td>
</tr>
</tbody>
</table>

## Virtual Machine Object

### Configuration

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add existing disk</td>
<td>VirtualMachine.Config.AddExistingDisk</td>
</tr>
<tr>
<td>Add new disk</td>
<td>VirtualMachine.Config.AddNewDisk</td>
</tr>
<tr>
<td>Add or remove device</td>
<td>VirtualMachine.Config.AddRemoveDevice</td>
</tr>
<tr>
<td>Advanced</td>
<td>VirtualMachine.Config.AdvancedConfig</td>
</tr>
<tr>
<td>Change CPU count</td>
<td>VirtualMachine.Config.CPUCount</td>
</tr>
<tr>
<td>Configure managedBy</td>
<td>VirtualMachine.Config.ManagedBy</td>
</tr>
<tr>
<td>Disk change tracking</td>
<td>VirtualMachine.Config.ChangeTracking</td>
</tr>
<tr>
<td>Disk lease</td>
<td>VirtualMachine.Config.DiskLease</td>
</tr>
<tr>
<td>Display connection settings</td>
<td>VirtualMachine.Config.MksControl</td>
</tr>
<tr>
<td>Extend virtual disk</td>
<td>VirtualMachine.Config.DiskExtend</td>
</tr>
<tr>
<td>Memory</td>
<td>VirtualMachine.Config.Memory</td>
</tr>
<tr>
<td>Modify device settings</td>
<td>VirtualMachine.Config.EditDevice</td>
</tr>
<tr>
<td>Raw device</td>
<td>VirtualMachine.Config.RawDevice</td>
</tr>
<tr>
<td>Reload from path</td>
<td>VirtualMachine.Config.ReloadFromPath</td>
</tr>
<tr>
<td>Remove disk</td>
<td>VirtualMachine.Config.RemoveDisk</td>
</tr>
<tr>
<td>Rename</td>
<td>VirtualMachine.Config.Rename</td>
</tr>
<tr>
<td>Reset guest information</td>
<td>VirtualMachine.Config.ResetGuestInfo</td>
</tr>
</tbody>
</table>
Guest Operations

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest Operation Program Execution</td>
<td>VirtualMachine.GuestOperations.Execute</td>
</tr>
<tr>
<td>Guest Operation Modifications</td>
<td>VirtualMachine.GuestOperations.Modify</td>
</tr>
<tr>
<td>Guest Operation Queries</td>
<td>VirtualMachine.GuestOperations.Query</td>
</tr>
</tbody>
</table>

Interaction

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer question</td>
<td>VirtualMachine.Interact.AnswerQuestion</td>
</tr>
<tr>
<td>Configure CD media</td>
<td>VirtualMachine.Interact.SetCDMedia</td>
</tr>
<tr>
<td>Device connection</td>
<td>VirtualMachine.Interact.DeviceConnection</td>
</tr>
<tr>
<td>Guest operating system management by VIX API</td>
<td>VirtualMachine.Interact.GuestControl</td>
</tr>
<tr>
<td>Power on</td>
<td>VirtualMachine.Interact.PowerOn</td>
</tr>
<tr>
<td>Reset</td>
<td>VirtualMachine.Interact.Reset</td>
</tr>
<tr>
<td>Suspend</td>
<td>VirtualMachine.Interact.Suspend</td>
</tr>
<tr>
<td>VMware Tools install</td>
<td>VirtualMachine.Interact.ToolsInstall</td>
</tr>
</tbody>
</table>

Inventory

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create from existing</td>
<td>VirtualMachine.Inventory.CreateFromExisting</td>
</tr>
<tr>
<td>Create new</td>
<td>VirtualMachine.Inventory.Create</td>
</tr>
<tr>
<td>Move</td>
<td>VirtualMachine.Inventory.Move</td>
</tr>
<tr>
<td>Remove</td>
<td>VirtualMachine.Inventory.Delete</td>
</tr>
</tbody>
</table>

Provisioning

When cloning a stemcell, BOSH sets custom specifications, such as hostnames and network configurations, based on the stemcell operating system.

The VM download privilege allows BOSH to modify files within a VM, including links between VMs and persistent disks. When vMotion migrates disks in vSphere, BOSH uses these links to maintain the connections between VMs and their persistent disks.

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow read-only disk access</td>
<td>VirtualMachine.Provisioning.DiskRandomRead</td>
</tr>
<tr>
<td>Allow virtual machine download</td>
<td>VirtualMachine.Provisioning.GetVmFiles</td>
</tr>
<tr>
<td>Allow virtual machine files upload</td>
<td>VirtualMachine.Provisioning.PutVmFiles</td>
</tr>
<tr>
<td>Clone template</td>
<td>VirtualMachine.Provisioning.CloneTemplate</td>
</tr>
<tr>
<td>Clone virtual machine</td>
<td>VirtualMachine.Provisioning.Clone</td>
</tr>
<tr>
<td>Customize</td>
<td>VirtualMachine.Provisioning.Customize</td>
</tr>
</tbody>
</table>
### Deploy template

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark as template</td>
<td>VirtualMachine.Provisioning.MarkAsTemplate</td>
</tr>
<tr>
<td>Mark as virtual machine</td>
<td>VirtualMachine.Provisioning.MarkAsVM</td>
</tr>
<tr>
<td>Modify customization specification</td>
<td>VirtualMachine.Provisioning.ModifyCustSpecs</td>
</tr>
<tr>
<td>Promote disks</td>
<td>VirtualMachine.Provisioning.PromoteDisks</td>
</tr>
<tr>
<td>Read customization specifications</td>
<td>VirtualMachine.Provisioning.ReadCustSpecs</td>
</tr>
</tbody>
</table>

### Snapshot Management

Before Ops Manager deploys a new VM, it uses a snapshot to clone the stemcell image to the destination.

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create snapshot</td>
<td>VirtualMachine.State.CreateSnapshot</td>
</tr>
<tr>
<td>Remove snapshot</td>
<td>VirtualMachine.State.RemoveSnapshot</td>
</tr>
<tr>
<td>Rename snapshot</td>
<td>VirtualMachine.State.RenameSnapshot</td>
</tr>
<tr>
<td>Revert snapshot</td>
<td>VirtualMachine.State.RevertToSnapshot</td>
</tr>
</tbody>
</table>

### vApp Object

These privileges must be set at the resource pool level. `VApp.ApplicationConfig` is required when attaching or detaching persistent disks.

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import</td>
<td>VApp.Import</td>
</tr>
<tr>
<td>vApp application configuration</td>
<td>VApp.ApplicationConfig</td>
</tr>
</tbody>
</table>
Deploying Ops Manager on vSphere

This topic provides instructions for deploying Pivotal Cloud Foundry (PCF) Ops Manager to VMware vSphere.

The instructions in this topic are based on vSphere 6.5 using the vSphere Client (HTML5) interface.

Runtime Considerations

For security purposes, VMware requires a dedicated instance of Ops Manager for use with vSphere.

Warning: If you are installing on vSphere with NSX-T integration, follow the instructions in Deploying Ops Manager with NSX-T for instead of performing the steps described in this topic.

Prepare vSphere

1. Log into vCenter using the vSphere Client (HTML5).
2. Select the vSphere Datacenter where you are deploying Ops Manager.
3. Create one or more vSphere clusters. A vSphere cluster is a collection of ESXi hosts and associated virtual machines (VMs) with shared resources and a shared management interface. Follow the steps below to create a vSphere cluster:

   a. Right-click the vSphere Datacenter object.
   b. Select New Cluster.
   c. Name the Cluster. For example, Cluster1.
   d. Select Turn ON vSphere DRS to enable DRS. Set DRS Automation to Partially Automated or Full Automated.
   e. Select Turn ON vSphere HA to enable vSphere HA.

4. If you created more than one vSphere cluster, configure shared storage for the clusters. You can allocate shared storage to the clusters using one of two common approaches: horizontal or vertical. For more information about configuring shared storage for clusters, see Storage Design in vSphere Reference Architecture.

   Warning: Failure to properly configure shared storage for vSphere clusters could result in downtime and unrecoverable loss of data.

5. Create a resource pool for the management components in your deployment. Resource pools are vSphere objects used to organize resources.

   Note: For highly available, production-ready environments, you must create more than one cluster. For more information, see Compute and HA Considerations and Scaling and Capacity Management in vSphere Reference Architecture.

   a. Right-click the vSphere Datacenter object that you created.
   c. Name the resource pool. For example, RP-MGMT.
   d. Review the default CPU and Memory settings and adjust if necessary. This is not common.
   e. Click OK.

Deploy Ops Manager

Ops Manager for vSphere is provided as an OVA file. An OVA file is a template for a VM. To deploy Ops Manager on vSphere, you import the OVA file into your vSphere environment.
1. Download the Pivotal Cloud Foundry Operations Manager for vSphere OVA file from Pivotal Network. Use the Releases dropdown menu to select an Ops Manager release.

2. Right-click the management resource pool that you created and select Deploy OVF Template. Alternatively, from the vCenter menu, select Actions > Deploy OVF Template.

3. At the Select an OVF template screen, click Browse.

4. Select the Ops Manager OVA file that you downloaded and click Open.

5. Review template selection and click Next.

6. At the Select a name and folder screen:
   - Enter a name for the Ops Manager virtual machine, or use the default name.
   - Select the Datacenter object.
   - Click Next.

7. At the Select a compute resource screen, select the compute resource where the Ops Manager VM will run and click Next. Select the Resource Pool for the management components.

8. At the Review Details screen, confirm the configuration up to this point and click Next.

Note: Hardware virtualization must be turned off if your vSphere host does not support VT-X/EPT. Refer to vSphere Requirements in PCF on vSphere Requirements for more information.
9. At the Select Storage screen, select the virtual disk format.
   
a. Choose a Datastore.

   - If you use vSAN, select the vSAN datastore and apply the default storage policy.
   - If you use a dedicated disk, select the appropriate disk format for your deployment needs. If in doubt, choose Thin Provision. For more information about disk formats, see vSphere Virtual Disk Types.

   b. Click Next.

   Warning: Ops Manager requires a VM with at least 8 GB memory.

10. At the Select Networks screen, select the standard VM Network, and click Next.
11. At the Customize template screen, enter the following information, and click Next.

- **IP Address**: The IP address of the Ops Manager network interface. For example, 10.0.0.2.
- **Netmask**: The network mask for Ops Manager. For example, 255.255.255.0.
- **Default Gateway**: The default gateway for Ops Manager to use. For example, 10.0.0.1.
- **DNS**: One or more DNS servers for the Ops Manager VM to use. For example, 10.20.20.1.
- **NTP Servers**: The IP address of one or more NTP servers for Ops Manager. For example, 10.113.60.176.
- **Admin Password**: A default password for the ubuntu user.
- **Public SSH Key**: (Optional) Enter pre-existing public SSH key value to allow SSH access to the Ops Manager VM.

⚠️ **warning**: You must specify a default password, an SSH key, or both. If you do not have either of these fields configured, Ops Manager will not start.

- **Custom hostname**: The hostname for the Ops Manager VM. For example, ops-manager.
12. At the **Ready to complete** screen, review the configuration settings and click **Finish**. This action begins the OVA import and deployment process.
13. Use the **Recent Tasks** panel at the bottom of the vCenter dashboard to check the progress of the OVA import and deployment.

   If the import or deployment is unsuccessful, check the configuration for errors. Fix any errors and restart the deployment process.

**Start Ops Manager**

1. Create a DNS entry for the IP address that you used for Ops Manager. You must use this fully qualified domain name when you log into Ops Manager.

   **Note:** Ops Manager security features require you to create a fully qualified domain name to access Ops Manager during the initial configuration. For more information, see [PCF on vSphere Requirements](#).

2. Select the Ops Manager VM and click **Actions > Power > Power On**.

   Once the VM boots, the Ops Manager web interface is available at the IP address that you specified.

   **Note:** It is normal to experience a brief delay before the interface is accessible while the web server and VM start.

3. Navigate to the fully qualified domain of your Ops Manager in a web browser.

   **Note:** It is normal to experience a brief delay before the interface is accessible while the web server and VM start.

**Next Steps**

After you complete this procedure, follow the instructions in [Configuring BOSH Director on vSphere](#).
Configuring BOSH Director on vSphere

This topic describes how to configure the BOSH Director tile for VMware vSphere.

You can also perform the procedures in this topic using the Ops Manager API. For more information, see Using the Ops Manager API.

warning: If you are installing on vSphere with NSX-T integration, follow the instructions in Configuring BOSH Director with NSX-T for instead of performing the procedure described in this topic.

After you complete this procedure, follow the configuration instructions for the runtime that you choose to install.

For example:

- To install Pivotal Application Service (PAS), see Configuring PAS.
- To install, see Installing on vSphere.

See Installing Runtimes for more information about Pivotal Cloud Foundry (PCF) runtimes.

Prerequisite

Before you begin this procedure, you must complete all steps in Deploying Ops Manager to vSphere.

Step 1: Set Up Ops Manager Authentication

1. In a web browser, navigate to the fully qualified domain of your Ops Manager.
2. When Ops Manager starts for the first time, you must choose one of the following:
   - Internal Authentication: If you use Internal Authentication, Ops Manager maintains your user database.
   - SAML Identity Provider: If you use a SAML Identity Provider (IdP), an external identity server maintains your user database.
   - LDAP Server: If you use a LDAP Server, an external identity server maintains your user database.
Internal Authentication

1. When redirected to the Internal Authentication page, you must complete the following steps:
   1. Enter a **Username**, **Password**, and **Password confirmation** to create a user with administrative privileges.
   2. Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   3. If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in the Configuring Proxy Settings for the BOSH CPI topic.
   4. Read the **End User License Agreement**, and select the checkbox to accept the terms.
   5. Click **Setup Authentication**.

2. Log in to Ops Manager with the username and password you created in the previous step.


   You should be able to log in, and you should see the BOSH Director tile is present and ready for configuration, indicated by the orange color.
SAML Identity Provider

1. Log in to your IdP console and download the IdP metadata XML. Optionally, if your IdP supports metadata URL, you can copy the metadata URL instead of the XML.

2. Copy the IdP metadata XML or URL to the Ops Manager SAML Identity Provider login page.

3. Enter values for the fields listed below. Failure to provide values in these fields results in a 500 error.
   - **SAML admin group**: Enter the name of the SAML group that contains all Ops Manager administrators. This field is case-sensitive.
   - **SAML groups attribute**: Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.

4. Enter your **Decryption passphrase**. Read the **End User License Agreement**, and select the checkbox to accept the terms.

5. Your Ops Manager login page appears. Enter your username and password. Click **Login**.

6. Download your SAML Service Provider metadata (SAML Relying Party metadata) by navigating to the following URLs:
   - **6a.** Ops Manager SAML service provider metadata: [https://OPS-MAN-FQDN:443/uaa/saml/metadata](https://OPS-MAN-FQDN:443/uaa/saml/metadata)
   - **6b.** BOSH Director SAML service provider metadata: [https://BOSH-IP-ADDRESS:8443/saml/metadata](https://BOSH-IP-ADDRESS:8443/saml/metadata)

---

Note: The same IdP metadata URL or XML is applied for the BOSH Director. If you use a separate IdP for BOSH, copy the metadata XML or URL from that IdP and enter it into the BOSH IdP Metadata text box in the Ops Manager login page.
7. Configure your IdP with your SAML Service Provider metadata. Import the Ops Manager SAML provider metadata from Step 6a above to your IdP. If your IdP does not support importing, provide the values below.

- **Audience URI (SP Entity ID):** [https://OPS-MAN-FQDN:443/uaa](https://OPS-MAN-FQDN:443/uaa)
- **Name ID:** Email Address
- **SAML authentication requests are always signed**

8. Import the BOSH Director SAML provider metadata from Step 6b to your IdP. If the IdP does not support an import, provide the values below.

- **Single sign on URL:** [https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP](https://BOSH-IP:8443/saml/SSO/alias/BOSH-IP)
- **Audience URI (SP Entity ID):** [https://BOSH-IP:8443](https://BOSH-IP:8443)
- **Name ID:** Email Address
- **SAML authentication requests are always signed**

9. Return to the **BOSH Director** tile, and continue with the configuration steps below.

## LDAP Server

- **For Server URL**, enter the URL that points to your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - ldap://: This specifies that the LDAP server uses an unencrypted connection.
  - ldaps://: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or that you import a trusted certificate to the JVM truststore.

- **For LDAP Username and LDAP Password**, enter the LDAP Distinguished Name (DN) and the password for binding to the LDAP Server. Example DN: `cn=administrator,ou=Users,dc=example,dc=com`

  - **Note:** Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

  - **Warning:** Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- **For User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  - **Example:** For a domain named “cloud.example.com” typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`

- **For User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `{cn=Smith}` returns all objects with a common name equal to `Smith`.

  - **In the LDAP search filter string that you use to configure your runtime, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.**

  - **In addition to `cn`, other attributes commonly searched for and returned are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`**.
For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base:

```
ou=Groups,dc=example,dc=com
```

For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is `member={0}`.

For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example `mail`.

For **LDAP RBAC Admin Group Name**, enter the DN of the LDAP group you want to have admin permissions in Ops Manager.

From the dropdown, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:

- Automatically follow any referrals.
- Ignore referrals and return partial result.
- Throw exception for each referral and abort.

For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.

If you are using an **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

Read the **End User License Agreement**, and select the checkbox to accept the terms.

Select **Provide an admin client in the BOSH UAA**. You can use this to enable BOSH automation with scripts and tooling. For more information, see Provision Admin Client in Creating UAA Clients for BOSH Director.

Click **Setup Authentication**.

Return to the **BOSH Director** tile, and continue with the configuration steps below.

**Step 2: Configure vCenter**

1. Click the **BOSH Director for vSphere** tile.

2. Select **vCenter Config**. Configure the one or more vCenters that host your PCF foundation.

3. Enter the following information:

   - **Name**: A name that you provide for your vCenter configuration. This field is used to identify the datacenter configuration in Ops Manager if you are configuring multiple datacenters.
   - **vCenter Host**: The hostname of the vCenter that manages ESXi/vSphere.
   - **vCenter Username**: A vCenter username with create and delete privileges for virtual machines (VMs) and folders.
   - **vCenter Password**: The password for the vCenter user specified above.
   - **Datacenter Name**: The name of the datacenter as it appears in vCenter.
- **Virtual Disk Type**: The Virtual Disk Type to provision for all VMs. For guidance on selecting a virtual disk type, see [vSphere Virtual Disk Types](#).

  **Note**: The following fields do not allow whitespace. Including whitespace characters in datastore or datastore folder names causes an error.

- **Ephemeral Datastore Names (comma delimited)**: The names of the datastores that store ephemeral VM disks deployed by Ops Manager.
- **Persistent Datastore Names (comma delimited)**: The names of the datastores that store persistent VM disks deployed by Ops Manager.
4. Select a network configuration from one of the following:

- **Standard vCenter Networking**: This is the default option when upgrading Ops Manager.

⚠️ **warning**: If you intend to deploy using Flannel as the CNI (as opposed to NSX-T), then select this option to configure the BOSH Director. Do not choose **NSX Networking**. When you configure the tile later on, you will configure Flannel as the container network interface (CNI).

If you intend to deploy with NSX-T, see [Installing on vSphere with NSX-T Data Center](#).

- **NSX Networking**: Select this option to enable VMware NSX Network Virtualization for non-runtime such as Pivotal Application Service (PAS).
Configure NSX networking by entering the following information:

- **NSX Mode**: Select either NSX-V or NSX-T.
- **NSX Address**: The address of the NSX manager.
- **NSX Username**: The username to connect to the NSX manager.
- **NSX Password**: The password for the username specified above.
- **NSX CA Cert**: A CA certificate in PEM format that authenticates to the NSX server. If the NSX Manager generated a self-signed certificate, use the following command to retrieve the CA certificate using OpenSSL:

  ```bash
  openssl s_client -showcerts -connect NSX-MANAGER-ADDRESS:443 < /dev/null 2> /dev/null | openssl x509
  ```

  Where `NSX-MANAGER-ADDRESS` is the address of the NSX manager.

5. Configure the following folder names:

   - **VM Folder**: The vSphere datacenter folder where Ops Manager places VMs. Enter `[YOUR-DEPLOYMENT]_vms` where `YOUR-DEPLOYMENT` corresponds to a descriptive name for your deployment. For example, `my_pcf_vms`.
   - **Template Folder**: The vSphere datacenter folder where Ops Manager places VMs. Enter `[YOUR-DEPLOYMENT]_templates`. For example, `my_pcf_templates`.
   - **Disk path Folder**: The vSphere datastore folder where Ops Manager creates attached disk images. You must not nest this folder. Enter `[YOUR-DEPLOYMENT]_disk`. For example, `my_pcf_disk`.

   Note: After initial deployment, you cannot edit folder names.

6. Click **Save**.

7. (Optional) Click **Add vCenter Config** toward the top of the form to configure additional vCenters. Once you click **Save**, your multiple vCenter Configs are listed in the **vCenter Configs** pane. For more information about multiple vCenter configs, see Managing Multiple Data Centers.

**Step 3: Director Config Page**
1. In Ops Manager, select Director Config.

![Director Config]

2. In the NTP Servers (comma delimited) field, enter a comma-separated list of valid NTP servers.

   **Note:** The NTP server configuration only updates after VM recreation. Ensure that you select the Recreate all VMs checkbox if you modify the value of this field.

3. Leave the JMX Provider IP Address field blank.

4. Leave the Bosh HM Forwarder IP Address field blank.

   **Note:** Starting in PAS v2.0, BOSH-reported component metrics are available in the Loggregator Firehose by default. If you continue to use the BOSH HM Forwarder to consume these component metrics, you may receive duplicate data. To prevent this, leave the Bosh HM Forwarder IP Address field blank.

5. Select the Enable VM Resurrctor Plugin checkbox to enable the BOSH Resurrctor functionality and increase your runtime availability. If you intend to install PAS, see Using BOSH Resurrctor and vSphere HA for more information about the Resurrctor plugin.

6. Select Enable Post Deploy Scripts to run a post-deploy script after deployment. This script allows the job to execute additional commands against a deployment.

   **Note:** If you intend to install, you must enable post-deploy scripts.

7. Select Recreate all VMs to force BOSH to recreate all VMs on the next deploy. This process does not destroy any persistent disk data.

8. Select Recreate All Persistent Disks to force BOSH to migrate and recreate persistent disks for the BOSH Director and all tiles. This process does not destroy any persistent disk data.

9. Select Enable bosh deploy retries to instruct Ops Manager to retry failed BOSH operations up to five times.

10. Select Skip Director Drain Lifecycle to prevent drain scripts from running when the BOSH Director is recreated.

11. Select Store BOSH Job Credentials on tmpfs (beta) to store credentials for BOSH jobs on temporary file storage (tmpfs) memory, rather than on disk. You must recreate all VMs for this setting to take effect.

12. (Optional) Disable Allow Legacy Agents if all of your tiles have stemcells v3468 or later. Disabling the field will allow Ops Manager to implement TLS...
secure communications.

13. Select **Keep Unreachable Director VMs** if you want to preserve BOSH Director VMs after a failed deployment for troubleshooting purposes.


   ![HM Pager Duty Plugin](image)

   - **Service Key**: Enter your API service key from PagerDuty.
   - **HTTP Proxy**: Enter an HTTP proxy for use with PagerDuty.

15. Select **HM Email Plugin** to enable Health Monitor integration with email.

   ![HM Email Plugin](image)

   - **Host**: Enter your email hostname.
   - **Port**: Enter your email port number.
   - **Domain**: Enter your domain.
   - **From**: Enter the address for the sender.
   - **Recipients**: Enter comma-separated addresses of intended recipients.
   - **Username**: Enter the username for your email server.
   - **Password**: Enter the password for your email server.
   - **Enable TLS**: Select this checkbox to enable Transport Layer Security to the email host.

16. For **CredHub Encryption Provider**, you can choose whether BOSH CredHub stores its encryption key internally on the BOSH Director and CredHub VM, or in an external hardware security module (HSM). The HSM option is more secure.

   Before configuring an HSM encryption provider in the **Director Config** pane, you must follow the procedures and collect information described in **Preparing CredHub HSMs for Configuration**.
Internal: Select this option for internal CredHub key storage. This option is selected by default and requires no additional configuration.

Luna HSM: Select this option to use a SafeNet Luna HSM as your permanent CredHub encryption provider, and fill in the following fields:

1. **Encryption Key Name**: Any name to identify the key that the HSM uses to encrypt and decrypt the CredHub data. Changing this key name after you deploy Ops Manager can cause service downtime.
2. **Provider Partition**: The partition that stores your encryption key. Changing this partition after you deploy Ops Manager could cause service downtime. For this value and the ones below, use values gathered in [Preparing CredHub HSMs for Configuration](#).
3. **Provider Partition Password**
4. **Provider Client Certificate**: The certificate that validates the identity of the HSM when CredHub connects as a client.
5. **Provider Client Certificate Private Key**
6. **HSM Host Address**
7. **HSM Port Address**: If you do not know your port address, enter 1792.
8. **Partition Serial Number**

**Note:** After you deploy Ops Manager with an HSM encryption provider, you cannot change BOSH CredHub to store encryption keys internally.
9. **HSM Certificate**: The certificate that the HSM presents to CredHub to establish a two-way mTLS connection.

17. **Select a Blobstore Location** to either configure the blobstore as an internal server or an external endpoint. Because the internal server is unscalable and less secure, Pivotal recommends that you configure an external blobstore.

**Note**: After you deploy Ops Manager, you cannot change the blobstore location.

- **Internal**: Select this option to use an internal blobstore. Ops Manager creates a new VM for blob storage. No additional configuration is required.
- **Enable TLS**: Select this checkbox to enable TLS to the blobstore.
  
  **Note**: If you are using PASW 2016, make sure you have downloaded Windows stemcell v1709.10 or higher before enabling TLS.

- **S3 Compatible Blobstore**: Select this option to use an external S3-compatible endpoint. Follow the procedures in [Sign up for Amazon S3](#) and [Creating a Bucket](#) in the AWS documentation. When you have created an S3 bucket, complete the following steps:

  1. **S3 Endpoint**: Navigate to the [Regions and Endpoints](#) topic in the AWS documentation.

     a. Locate the endpoint for your region in the [Amazon Simple Storage Service (S3)](#) table and construct a URL using your region’s...
endpoint. For example, if you are using the us-west-2 region, the URL you create would be
https://s3-us-west-2.amazonaws.com. Enter this URL into the S3 Endpoint field.

b. On a command line, run ssh ubuntu@OPS-MANAGER-FQDN to SSH into the Ops Manager VM. Replace OPS-MANAGER-FQDN with the fully qualified domain name of Ops Manager.

c. Copy the custom public CA certificate you used to sign the S3 endpoint into /etc/ssl/certs on the Ops Manager VM.

d. On the Ops Manager VM, run sudo update-ca-certificates -f -v to import the custom CA certificate into the Ops Manager VM truststore.

2. **Bucket Name**: Enter the name of the S3 bucket.
3. **Access Key** and **Secret Key**: Enter the keys you generated when creating your S3 bucket.
4. **Select V2 Signature or V4 Signature**: If you select V4 Signature, enter your **Region**.

Note: AWS recommends using Signature Version 4. For more information about AWS S3 Signatures, see Authenticating Requests in the AWS documentation.

- **GCS Blobstore**: Select this option to use an external GCS endpoint. To create a GCS bucket, you must have a GCS account. Follow the procedures in Creating Storage Buckets in the GCS documentation to create a GCS bucket. When you have created a GCS bucket, complete the following steps:

  1. **Bucket Name**: Enter the name of your GCS bucket.
  2. **Storage Class**: Select the storage class for your GCS bucket. See Storage Classes in the GCP documentation for more information.
  3. **Service Account Key**: Follow the steps in the Set up IAM Service Accounts section of Preparing to Deploy Ops Manager on GCP Manually to download a JSON file with a private key. Enter the contents of the JSON file into the field.

18. **Select a Database Location**. By default, Ops Manager deploys and manages an **Internal** database for you. If you choose to use an **External MySQL Database**, complete the associated fields with information obtained from your external MySQL Database provider: **Host**, **Port**, **Username**, **Password**, and **Database**.

In addition, if you selected the **Enable TLS for Director Database** checkbox, you can complete the following optional fields:

- **Enable TLS**: Select this checkbox enables TLS communication between the BOSH Director and the database.
- **TLS CA**: Enter the Certificate Authority for the TLS Certificate.
- **TLS Certificate**: Enter the client certificate for mutual TLS connections to the database.
- **TLS Private Key**: Enter the client private key for mutual TLS connections to the database.
- **Advanced DB Connection Options**: If you would like to provide additional options for the database, use this field to provide a JSON-formatted options string.

19. **(Optional)** Modify the **Director Workers** value, which sets the number of workers available to execute Director tasks. This field defaults to **3**.

20. **(Optional)** **Max Threads** sets the maximum number of threads that the BOSH Director can run simultaneously. Pivotal recommends that you leave
the field blank to use the default value, unless doing so results in rate limiting or errors on your IaaS.

21. (Optional) To add a custom URL for your BOSH Director, enter a valid hostname in Director Hostname. You can also use this field to configure a load balancer in front of your BOSH Director. For more information, see How to Set Up a Load Balancer in Front of Operations Manager Director in the Pivotal Support Knowledge Base.

22. (Optional) To set a custom banner that users see when logging in to the Director using SSH, enter text in the Custom SSH Banner field.

23. (Optional) Enter your comma-separated custom Identification Tags. For example, iaas:foundation1, hello:world. You can use the tags to identify your foundation when viewing VMs or disks from your IaaS.

24. Click Save.

Step 4: Create Availability Zones Page

Ops Manager Availability Zones (AZs) correspond to your vCenter clusters and resource pools.

Multiple AZs allow you to provide high availability and load balancing to your applications. When you run more than one instance of an application, Ops Manager balances those instances across all of the AZs assigned to the application.

Pivotal recommends that you use at least three AZs for a highly available installation of your chosen runtime.

1. Select Create Availability Zones to navigate to the pane.
2. Use the following steps to create the first AZ.

- Click **Add**.
- Enter a unique **Name** for the AZ.
- Select a vCenter config name from the **IaaS Configuration** dropdown to associate your AZ with a vCenter. If you have only one vCenter config, or if you are creating your first AZ, IaaS Configuration defaults to the first vCenter and cannot be configured.
- Enter the name of the vCenter **Cluster** where you deployed Ops Manager. You created this cluster in the **Prepare vSphere** step of Deploying Ops Manager to vSphere. For example, Cluster1.
- Enter the name of the **Resource Pool** that you created in the **Prepare vSphere** step of Deploying Ops Manager to vSphere to organize management components like Ops Manager and BOSH Director. For example, RP-MGMT. The jobs running in this AZ share the CPU and memory resources defined by the pool. You only need to specify this resource group in one AZ.
- (Optional) Enter the name of a **Host Group** for the AZ to use. The host group must be within the cluster you designated in the **Clusters** field.
- (Optional) Click **Add Cluster** to add more clusters to the AZ as necessary for high availability requirements.

3. (Optional) Use the following steps to create additional AZs.

- Click **Add**.
- Specify **Cluster** and optionally, a **Resource Pool**.
- Click **Add Cluster** to assign more clusters to the AZ as necessary for high availability. Click the trash icon to delete a cluster. The first cluster cannot be deleted.

4. Click **Save**.

**Step 5: Create Networks Page**

1. Select **Create Networks**.

2. Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are
reachable.

3. Depending on the runtime that you are deploying, do one of the following:
   - For PAS, click Add Network and create the following networks:
     - **infrastructure** : This network is for Ops Manager and the BOSH Director.
     - **pas** : This network is for all the PAS VMs including Gorouter, Diego cells, and Cloud Controller.
     - **services** : This network is for any service tiles to be deployed alongside PAS.

   Use the values from the following table as a guide when you create each network, replacing the IP addresses with ranges that are available in your vSphere environment:

<table>
<thead>
<tr>
<th>Field</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>infrastructure</td>
</tr>
<tr>
<td>vSphere Network Name</td>
<td>pcf-virt-net/infrastructure</td>
</tr>
<tr>
<td>CIDR</td>
<td>192.168.101.0/24</td>
</tr>
<tr>
<td>DNS</td>
<td>192.168.101.2</td>
</tr>
<tr>
<td>Gateway</td>
<td>192.168.101.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>pas</td>
</tr>
<tr>
<td>vSphere Network Name</td>
<td>pcf-virt-net/pcf-pas-subnet</td>
</tr>
<tr>
<td>CIDR</td>
<td>192.168.16.0/24</td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>192.168.16.1-192.168.16.9</td>
</tr>
<tr>
<td>DNS</td>
<td>192.168.16.2</td>
</tr>
<tr>
<td>Gateway</td>
<td>192.168.16.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>services</td>
</tr>
<tr>
<td>vSphere Network Name</td>
<td>pcf-virt-net/pcf-services-subnet</td>
</tr>
<tr>
<td>CIDR</td>
<td>192.168.20.0/24</td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>192.168.20.1-192.168.20.9</td>
</tr>
<tr>
<td>DNS</td>
<td>192.168.20.2</td>
</tr>
<tr>
<td>Gateway</td>
<td>192.168.20.1</td>
</tr>
</tbody>
</table>

   * In **vSphere Network Name**, enter the full path of the network as it displays in vCenter. For example, enter `YOUR-DIRECTORY-NAME/YOUR-NETWORK-NAME`. If your vSphere Network Name contains a forward slash character, replace the forward slash with the URL-encoded forward slash character `%2f`.

   † For **Reserved IP Ranges**, enter any IP addresses from the **CIDR** that you want to blacklist from the installation. Ops Manager will not deploy VMs to any address in this range.

   - For , click Add Network and create the following networks:
     - **infrastructure** : This network is for Ops Manager, the BOSH Director, the PKS broker, and the PKS API.
     - **pks** : If you have a large deployment with multiple tiles, you can choose to deploy the broker and API to a separate network named `pks`. See the table below for more information.
     - **services** : Network for creating the master and worker VMs for Kubernetes clusters. The CIDR should not conflict with the pod overlay network `10.200.0.0/16` or the reserved Kubernetes services CIDR of `10.100.200.0/24`.

   ![](Note:
   Multiple networks allow you to place vCenter on a private network and the rest of your deployment on a public network. Isolating vCenter in this manner denies access to it from outside sources and reduces possible security vulnerabilities.

   Use the values from the following table as a guide when you create each network, replacing the IP addresses with ranges that are available in your vSphere environment:

<table>
<thead>
<tr>
<th>Field</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>infrastructure</td>
</tr>
<tr>
<td>vSphere Network Name</td>
<td>pcf-virt-net/pks-infrastructure-subnet</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
<table>
<thead>
<tr>
<th><strong>Infrastructure Network</strong></th>
<th><strong>CIDR</strong></th>
<th>192.168.101.0/26</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reserved IP Ranges</strong> †</td>
<td>192.168.101.1-192.168.101.9</td>
<td></td>
</tr>
<tr>
<td><strong>DNS</strong></td>
<td>192.168.101.2</td>
<td></td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>192.168.101.1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Main Network (Optional)</strong></th>
<th><strong>Field</strong></th>
<th><strong>Configuration Example</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>pks</td>
<td></td>
</tr>
<tr>
<td><strong>vSphere Network Name</strong> †</td>
<td>pcf-virt-net/pks-subnet</td>
<td></td>
</tr>
<tr>
<td><strong>CIDR</strong></td>
<td>192.168.16.0/26</td>
<td></td>
</tr>
<tr>
<td><strong>Reserved IP Ranges</strong> †</td>
<td>192.168.16.1-192.168.16.9</td>
<td></td>
</tr>
<tr>
<td><strong>DNS</strong></td>
<td>192.168.16.2</td>
<td></td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>192.168.16.1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Service Network</strong></th>
<th><strong>Field</strong></th>
<th><strong>Configuration Example</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>services</td>
<td></td>
</tr>
<tr>
<td><strong>vSphere Network Name</strong> †</td>
<td>pcf-virt-net/pks-service-subnet</td>
<td></td>
</tr>
<tr>
<td><strong>CIDR</strong></td>
<td>192.168.20.0/22</td>
<td></td>
</tr>
<tr>
<td><strong>Reserved IP Ranges</strong> †</td>
<td>192.168.20.1-192.168.20.9</td>
<td></td>
</tr>
<tr>
<td><strong>DNS</strong></td>
<td>192.168.20.2</td>
<td></td>
</tr>
<tr>
<td><strong>Gateway</strong></td>
<td>192.168.20.1</td>
<td></td>
</tr>
</tbody>
</table>

† In **vSphere Network Name**, enter the full path of the network as it displays in vCenter. For example, enter `YOUR-DIRECTORY-NAME/YOUR-NETWORK-NAME`. If your vSphere Network Name contains a forward slash character, replace the forward slash with the URL-encoded forward slash character `%2f`.

† For **Reserved IP Ranges**, enter any IP addresses from the CIDR that you want to blacklist from the installation. Ops Manager will not deploy VMs to any address in this range.

4. For each network that you create, select the **Availability Zones** to use with the network. Assign as many AZs to your network as needed by your deployment. For more information, see Compute and HA Considerations and Scaling and Capacity Management sections of the vSphere Reference Architecture.

5. Click **Save**.

**Step 6: Assign AZs and Networks**

1. Select **Assign AZs and Networks**.

   ![Assign AZs and Networks](image)

2. Use the dropdown to select a **Singleton Availability Zone**. The BOSH Director installs as a single instance in this Availability Zone.

3. Use the dropdown to select a **Network** for your BOSH Director.
4. Click **Save**.

**Step 7: Security Page**

1. Select **Security**.

![Security Page](image)

2. In **Trusted Certificates**, enter your custom certificate authority (CA) certificates to insert into your organization's certificate trust chain. This feature enables all BOSH-deployed components in your deployment to trust custom root certificates.

To enter multiple certificates, paste your certificates one after the other. For example, format your certificates like the following:

```
-----BEGIN CERTIFICATE-----
ABCDEF0123456789ABCD123456789ABCDEF0123456789
EFCH123456789ABCD123456789ABCDEF123456789ABCDEF
GHI123456789ABCD123456789ABCDEF123456789ABCDEF
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
BCDEFG0123456789ABCD123456789ABCDEF0123456789
EFCH123456789ABCD123456789ABCDEF123456789ABCDEF
GHI123456789ABCD123456789ABCDEF123456789ABCDEF
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
CDEFGH0123456789ABCD123456789ABCDEF0123456789
EFCH123456789ABCD123456789ABCDEF123456789ABCDEF
GHI123456789ABCD123456789ABCDEF123456789ABCDEF
-----END CERTIFICATE-----

**Note:** If you want to use Docker registries to run PAS app instances in Docker containers, enter the certificate for your private Docker registry in this field. See [Using Docker Registries](#) for more information about running app instances in PAS using Docker registries.

3. (Optional) Select the **Include OpsManager Root CA in Trusted Certs** checkbox to include the Ops Manager root CA in the Trusted Certificates field. BOSH Director includes this CA in the trust store of every VM that it deploys.

4. Choose **Generate passwords** or **Use default BOSH password**. Pivotal recommends that you use the **Generate passwords** option for greater security.

5. Click **Save**. To view your saved Director password, click the **Credentials** tab.
Step 8: BOSH DNS Config Page

1. Select BOSH DNS Config.

2. (Optional) In Excluded Recursors, enter a list of prohibited recursor addresses.

3. (Optional) In Recursor Timeout, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   📘 Note: This time limit must include one of the Go parse duration time units. For example, entering $5s$ sets the timeout limit to five seconds. For more information about supported time units, see `func ParseDuration` in the Go Programming Language documentation.

4. (Optional) In Handlers, enter a list of custom domain handlers in JSON format.

5. Click Save.

Step 9: Syslog Page

📘 Note: BOSH Director logs contain sensitive information that should be considered privileged. For example, these logs may contain cloud provider credentials. If you choose to forward logs to an external syslog endpoint, use TLS encryption to prevent information from being intercepted by a third party.
1. Select Syslog.

2. (Optional) Select Yes to send BOSH Director system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in Address.

4. Enter the port number that the remote server listens on in Port.

5. Select TCP or UDP from the Transport Protocol dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the Enable TLS checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in Permitted Peer.
   b. Enter the SSL certificate for the remote server in SSL Certificate.
7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** field. This field requires the rainier.script syntax.

10. Click **Save Syslog Settings**.

### Step 10: Resource Config Page

1. Select **Resource Config**.

<table>
<thead>
<tr>
<th>JOB</th>
<th>INSTANCES</th>
<th>PERSISTENT DISK TYPE</th>
<th>VM TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager Director</td>
<td>Automatic: 1</td>
<td>Automatic: 5 GB</td>
<td>Automatic: medium.disk (cpu: 2, ram: 4 GB)</td>
</tr>
<tr>
<td>Master Compilation Job</td>
<td>Automatic: 4</td>
<td>None</td>
<td>Automatic: large.cpu (cpu: 4, ram: 4 GB)</td>
</tr>
</tbody>
</table>

2. Adjust any values as necessary for your deployment. Under the **Instances**, **Persistent Disk Type**, and **VM Type** fields, choose **Automatic** from the dropdown to allocate the recommended resources for the job. If the **Persistent Disk Type** field reads **None**, the job does not require persistent disk space.

- **Note:** Ops Manager requires a Director VM with at least 8 GB memory.

- **Note:** If you set a field to **Automatic** and the recommended resource allocation changes in a future version, Ops Manager automatically uses the updated recommended allocation.

- **Note:** If you install PAS for Windows, provision your **Master Compilation Job** with at least 100 GB of disk space.

3. Click **Save**.

### Step 11: (Optional) Add Custom VM Extensions

Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers. For more information, see [Managing Custom VM Extensions](#).

### Step 12: Complete the BOSH Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Review Pending Changes**. Select the product that you intend to deploy and review the changes. For more information, see [Reviewing Pending Product Changes](#).
Next Steps

After you complete this procedure, follow the instructions for the runtime that you intend to install.

- To install PAS, see Configuring PAS.
- To install, see Installing on vSphere.

For more information about PCF runtimes, see Installing Runtimes.
Deploying PAS with NSX-T Networking

This topic describes how to install Pivotal Application Service (PAS) on vSphere with NSX-T internal networking, using the VMware NSX-T Container Plug-in for PCF.

Overview

PAS v2.4 uses a Container Network Interface (CNI) plugin to support secure and direct internal communication between containers. This plugin can be:

- The internal Silk plugin that comes packaged with PAS, or
- On vSphere, VMware NSX-T Container Plug-in for PCF, which installs as an Ops Manager tile.

Prerequisites

Before deploying PAS with NSX-T networking, you must have the following:

- An NSX-T 2.2 or later environment with NSX-T components installed and configured. See the NSX-T Cookbook for directions.
- BOSH and Ops Manager v2.3 or later installed and configured on vSphere. See Deploying Ops Manager on vSphere and Configuring BOSH Director on vSphere.
- The VMware NSX-T Container Plug-in for PCF tile downloaded from Pivotal Network and imported to the Ops Manager Installation Dashboard. See Adding and Importing Products for how to download and import Pivotal products to the Installation Dashboard.
- The PAS tile downloaded from Pivotal Network and imported to the Ops Manager Installation Dashboard. The PAS tile must be in one of the following two states:
  - Not deployed yet; you have not yet clicked Review Pending Changes, then Apply Changes on this version of PAS.
  - Deployed previously, with the Networking pane > Container Network Interface Plugin set to External.

Note: Deploying PAS with its Container Network Interface (CNI) set to Silk configures Diego cells to use an internally-managed container network. Subsequently switching the CNI interface to External NSX-T leads to errors.

Architecture

The following diagram shows how to deploy an NSX-T machine to run PAS across multiple vSphere hardware clusters. NSX-T runs a Tier-0 (T0) router and multiple Tier-1 (T1) routers, each connecting to a network within Pivotal Cloud Foundry. Each vSphere hardware column cluster corresponds to an Availability Zone in PCF.
When a developer pushes an app to a new Org for the first time, the NSX-T plugin triggers NSX-T to create a new T1 router and allocate an address range for the Org, on demand.

Install and Configure PAS and NSX-T

Installing NSX-T to run with PAS requires the following actions, which are described below:

1. Set up NSX-T to Integrate with PAS
2. Enable NSX-T Mode in the BOSH Director
3. Configure PAS for External Container Networking
4. Install and Configure the NSX-T Tile

Set up NSX-T to Integrate with PAS

To set up NSX-T to integrate with PAS, complete the procedures in the following sections:

- Set up Logical Switches
- Set up Routers
- Set up Load Balancer

Set up Logical Switches
To set up logical switches, do the following:

1. In vSphere, create logical network switches to correspond to the networks that PCF uses.
   a. Log into the NSX-T Manager Dashboard.
   b. Open the Advanced Networking & Security tab at the top.
   c. Open the Switching tab on the left.
   d. For each of these networks...

   - **Infrastructure** (BOSH and Ops Manager, defined in BOSH Director tile > Assign AZs and Networks pane)
   - **Deployment** (PAS, defined in PAS tile > Assign AZs and Networks pane)
   - **Services and Dynamic Services** (marketplace services and on-demand services, also defined in the PAS tile)
   - **Isolation Segment** (optional, defined in the Isolation Segment tile > Assign AZs and Networks pane) …do the following:

   1. Click +ADD.
   2. Enter a name for the logical switch (e.g. `PAS-Infrastructure`, `PAS-Deployment`).
   3. Click ADD.

![Add New Logical Switch](image)
To set up routers, do the following:

1. Create T0 network address translation (NAT) rules to communicate with Ops Manager
   a. Open the Advanced Networking & Security tab at the top
   b. Open the Routers tab on the left
   c. Select your T0 router
   d. Select the Services drop-down menu and choose NAT.
   e. Add a rule for destination NAT (DNAT) with:
      - The externally-specified destination IP address of incoming requests. If your Ops Manager has a DNS entry (e.g., `opsmgr.example.com`), this would be its IP address.
      - The internal network address of the Ops Manager

   ![New NAT Rule](image)

   f. Add the corresponding source NAT (SNAT) rule with:
      - The externally-specified destination IP address
      - The internal network address of the Ops Manager
g. Add a rule for source NAT (SNAT) for the infrastructure and deployment networks:

- The externally-specified destination IP address
- The internal network address in CIDR notation
2. Create T1 Routers for PAS, to connect from the T0 router. For each PCF network, Infrastructure, Deployment, and so on, create a T1 router as follows:

3. In the NSX-T Manager UI, navigate to Advanced Networking & Security > Routing > Routers
   a. Click +ADD > Tier-1 Router.
   b. Configure the router. Include the Edge Cluster and Edge Cluster Members; they are required to enable the Load Balancer. The Infrastructure network router configuration might look like:
4. Create T1 router downlink ports for PAS. For each T1 router you created, add a New Router Port as follows, to allow traffic in and out:
   a. In the NSX-T Manager UI, select the T1 router.
   b. In Configuration > Router Ports, click +ADD to add a new router port.
   c. For Logical Switch, enter the name of the logical switch you defined for the network in Add New Logical Switch, above.
   d. For IP Address, use the first IP of the appropriate subnet. In this example, we have set aside 192.168.1.0/24 for Infrastructure (Operations Manager and BOSH Director), and 192.168.2.0/24 for the Deployment, so we used 192.168.1.1 and 192.168.2.1 respectively.
5. Advertise the routes of the T1 routers to the T0 router, so the T0 router can correctly route incoming requests based on their destination IP address:

   a. Select your T1 Router and navigate to Routing > Route Advertisement.
   b. Under Edit Route Advertisement Configuration, enable route advertisement by setting Status to Enabled.
   c. Set Advertise All Connected Routes to Yes.
   d. Set Advertise All LB VIP Routes to Yes (necessary if Load Balancing service is configured).
6. Allocate an IP Block for PAS Orgs.

   a. From the NSX-T Manager, navigate to Advanced Networking & Security > Networking > IPAM and click +ADD.
   b. Enter a name (e.g., PAS-container-ip-block). Later you will enter this IP Block name on the VMware NSX-T tile in the NCP section under IP Blocks of Container Networks.
   c. Enter a description, such as Subnets are allocated from this pool to each newly-created Org.
   d. Enter a CIDR to allocate an address block large enough to accommodate all PAS apps. A /14 CIDR is large enough for ~1,000 Orgs with ~250 apps each. If you’re planning such a large foundation, pay heed to the VMware NSX-T PAS limits.
7. Create an external SNAT IP pool.
   a. Navigate to Advanced Networking & Security > Inventory > Groups > IP Pools and click +ADD.
   b. Enter a name (e.g. `external-ip-pool`) and a description (e.g. IP pool that provides 1 public IP for each Cloud Foundry Org). Later you will enter this pool name on the VMware NSX-T tile in the NCP section under IP Pools used to provide External (NAT) IP Addresses to Org Networks.

Add New IP Pool

Name *

**external-ip-pool**

Description

IP pool that provides 1 public IP for each Cloud Foundry Org
Set up Load Balancer

To set up a load balancer, do the following:

1. Create Active Health Monitors (health checks) for use by the Virtual Server later on:
   a. In the NSX-T Manager UI, navigate to Advanced Networking & Security > Networking > Load Balancing > Monitors > Active Health Monitors
   b. Create the health monitor for web load balancing:
      c. Click +ADD
      d. Fill out Monitor Properties:
         ■ Name: pas-web-monitor
         ■ Health Check Protocol: LbHttpMonitor
         ■ Monitoring Port: 8080
      e. Click Next
      f. Fill out Health Check Parameters:
         ■ HTTP Method: GET
         ■ HTTP Request URL: /health
         ■ HTTP Response Code: 200
   g. Click Finish
   h. Create the health monitor for TCP load balancing:
      i. Click +ADD
      j. Fill out Monitor Properties:
         ■ Name: pas-tcp-monitor
         ■ Health Check Protocol: LbHttpMonitor
         ■ Monitoring Port: 80
   k. Click Next
   l. Fill out Health Check Parameters:
      ■ HTTP Method: GET
      ■ HTTP Request URL: /health
      ■ HTTP Response Code: 200
   m. Create the health monitor for SSH load balancing:
      n. Click +ADD
      o. Fill out Monitor Properties:
         ■ Name: pas-ssh-monitor
         ■ Health Check Protocol: LbTcpMonitor
         ■ Monitoring Port: 2222
   p. Click Next > Finish

2. Create Server Pools (collections of VMs which handle traffic) for use by the Virtual Server later on:
   a. In the NSX-T Manager UI, navigate to Advanced Networking & Security > Networking > Load Balancing > Server Pools
   b. Create the server pool for web load balancing:
      c. Click +ADD to add a new pool
      d. Fill out General Properties:
         ■ Name: pas-web-pool
      e. Click Next
      f. Fill out SNAT Translation:
         ■ Translation Mode: Auto Map
      g. Click Next
      h. Fill out Pool Members:
         ■ Membership Type: Static
      i. Click Next
      j. Fill out Health Monitors:
         ■ Active Health Monitor: pas-web-monitor
3. Create Virtual Servers:

   a. In the NSX-T Manager UI, navigate to Advanced Networking & Security > Networking > Load Balancing > Virtual Servers
   b. Create the Virtual Server which forwards unencrypted web (HTTP) traffic to the Foundation:

   ![Note: Foundations requiring end-to-end encryption should not enable the virtual server on port 80, or, if enabled, should configure it to redirect traffic to the encrypted port (443)]

   c. Click +ADD
   d. Fill out General Properties:
      - **Name**: pas-web-vs
      - **Application Type**: Layer 4 (TCP)
      - **Application Profile**: nsx-default-1b-fast-tcp-profile
   e. Click Next
   f. Fill out Virtual Server Identifiers:
      - **IP Address**: use the address of the DNS record of *.system.YOUR-SYSTEM-DOMAIN.com
      - **Port**: 80, 443
   g. Fill out Server Pool and Rules:
      - **Default Server Pool**: pas-web-pool
   h. Click Next several times, then Finish
Create the Virtual Server which forwards traffic to apps with custom ports to the Foundation:

Click +ADD to add a new Virtual Server

Fill out General Properties:

- **Name**: pas-tcp-vs
- **Application Type**: Layer 4 (TCP)
- **Application Profile**: nsx-default-lb-fast-tcp-profile

Click Next

**Virtual Server Identifiers**:

- **IP Address**: use the address of the DNS record of tcp.apps.YOUR-SYSTEM-DOMAIN.com
- **Port**: use the same ports as configured in the PAS Tile > Networking > TCP Routing Ports, e.g. 3024-1123,5900

Click Next

Fill out Server Pool and Rules:

- **Default Server Pool**: pas-tcp-pool

Click Next > Finish

Create the Virtual Server which forwards ssh traffic to the Foundation:

Click +ADD to add a new Virtual Server

Fill out General Properties:

- **Name**: pas-ssh-vs
- **Application Type**: Layer 4 (TCP)
- **Application Profile**: nsx-default-lb-fast-tcp-profile

Click Next

**Virtual Server Identifiers**:

- **IP Address**: use the address of the DNS record of ssh.system.YOUR-SYSTEM-DOMAIN.com
- **Port**: 2222

Click Next

Fill out Server Pool and Rules:

- **Default Server Pool**: pas-ssh-pool

Click Next > Finish

4. Create the Load Balancer:

a. In the NSX-T Manager UI, navigate to Advanced Networking & Security > Networking > Load Balancing > Load Balancers
b. Click +ADD
c. Fill out the fields:

- **Name**: pas-lb
- **Load Balancer Size**: Choose Small unless you have a larger Foundation
d. Click OK
e. Select pas-lb
f. Click Actions > Attach to a Virtual Server, and then select pas-web-vs. Repeat this procedure for the Virtual Servers pas-tcp-vs and pas-ssh-vs.
g. Click Action > Attach to a Logical Router, and then select T1-Router-PAS-Deployment

Enable NSX-T Mode in the BOSH Director

To enable NSX-T mode in the BOSH Director, do the following:

1. From the Ops Manager Installation Dashboard, open the BOSH Director tile.
2. In the vCenter Configs pane, click the pencil icon for the vCenter Config you want to edit. Then select NSX Networking.
3. Configure NSX networking by entering the following:

- **NSX Mode**: Select **NSX-T**.
- **NSX Address**: Enter the address of the NSX-T Manager.
- **NSX Username**: Enter the username to connect to the NSX-T Manager.
- **NSX Password**: The password for the username specified above.
- **NSX CA Cert**: A CA certificate in PEM format that authenticates to the NSX-T server. If the NSX-T Manager generated a self-signed certificate, you can retrieve the CA certificate using OpenSSL with the command
  ```bash
  openssl s_client -host NSX-ADDRESS -port 443 -prexit -showcerts
  ```

Configure PAS for External Container Networking

To configure PAS for external container networking, do the following:

1. If you have not already done so, download the PAS tile from Pivotal Network and import it to the Installation Dashboard. See Adding and Importing Products for directions.
2. Configure PAS, following the directions in Configuring PAS. When you configure Networking, select External under Container Networking Interface Plugin.

3. Update the server pool membership for the NSX-T Load Balancers by following Updating NSX-T Load Balancer Server Pool Membership. Make sure to make a VM extension for each of the three server pools: `pas-web-pool`, `pas-tcp-pool`, and `pas-ssh-pool`.

Install and Configure the NSX-T tile

To install and configure the NSX-T tile, do the following:

1. If you have not already done so, download the VMware NSX-T tile version 2.3 or later from Pivotal Network and import it to the Installation Dashboard. See Adding and Importing Products for directions.
2. Click the VMware NSX-T tile to open its Settings tab, and configure the NSX Manager pane as follows:

- **NSX Manager Address**: NSX-T Manager host address or IP address
- **Use Client Certificates or Username/Password**: Select Basic Authentication with Username and Password and enter NSX Manager Admin Username and Admin Password credentials in the fields underneath.
- **NSX Manager CA Cert**: Obtain this certificate from NSX-T Manager as follows:
  1. `ssh` into NSX-T Manager using the admin account that you created when you deployed NSX-T Manager.
  2. From the NSX-T Manager command line, run `get certificate api` to retrieve the certificate.

3. Open and configure the NCP (NSX-T Container Plugin) pane as follows:

- **PAS foundation Name**: If unsure, use `PAS`. If multiple foundations co-exist on the same NSX-T Manager, choose a unique string, e.g. `PAS-beta`. NCP creates artifacts (e.g. T1 routers) and prefixes their names with this string for easy identification.
- **Overlay Transport Zone**: A uniquely identifying string for the Transport Zone that you chose when you created logical switches for each network. This can be the name of the transport zone if no other zones in NSX-T share the same name, or else the UUID for the transport zone.
- **Tier-0 Router**: A uniquely identifying string for the T0 router. This can be the tag string that you gave the router in NSX-T Manager if no other T0 routers in NSX-T share the same name, or else the UUID for the router.
- **Subnet Prefix of Container Networks**: Subnet mask to set the address range size for apps in a single org. Defaults to `24`. This number must be higher than the mask for all PAS orgs in the NSX-T Manager New IP Block pane, to define a each Org’s fraction of the total PAS address space.
- **Enable SNAT for Container Network**: Enable this checkbox.
4. In the NSX Node Agent pane, enable the Enable Debug Level of Logging for NSX Node Agent checkbox.

5. Click Save and return to the Installation Dashboard.

6. After you have configured both the PAS tile and the VMware NSX-T tile, click Review Pending Changes, then Apply Changes to deploy PAS with NSX-T networking.

Automation

For more information about automation, see the following resources:

- Concourse Pipelines: Configure NSX-T for PAS: This sample Concourse pipeline provides jobs setup switches, routers, an IP block, and an IP pool to be used by PAS.
**PyNSXT** is a Python library that can be used as a CLI to run commands against a VMware NSX-T installation. It exposes operations for working with logical switches, logical routers, and pools. It uses version 2.1 of NSX-T for the swagger client.
vSphere Virtual Disk Types

When you create a virtual machine in VMware vSphere, vSphere creates a new virtual hard drive for that virtual machine. The virtual hard drive is contained in a virtual machine disk (VMDK). The disk format you choose for the new virtual hard drive can have a significant impact on performance.

You can choose one of three formats when creating a virtual hard drive:

- Thin Provisioned
- Thick Provisioned Lazy Zeroed
- Thick Provisioned Eager Zeroed

**Thin Provisioned**

**Advantages:**
- Fastest to provision
- Allows disk space to be over-committed to VMs

**Disadvantages:**
- Slowest performance due to metadata allocation overhead and additional overhead during initial write operations
- Over-commitment of storage can lead to application disruption or downtime if resources are actually used
- Does not support clustering features

When vSphere creates a thin provisioned disk, it only writes a small amount of metadata to the datastore. It does not allocate or zero out any disk space. At write time, vSphere first updates the allocation metadata for the VMDK, then zeros out the block or blocks, then finally writes the data. Because of this overhead, thin provisioned VMDKs have the lowest performance of the three disk formats.

Thin provisioning allows you to overcommit disk spaces to VMs on a datastore. For example, you could put 10 VMs, each with a 50 GB VMDK attached to it, on a single 100 GB datastore, as long as the sum total of all data written by the VMs never exceeded 100 GB. Thin provisioning allows administrators to use space on datastores that would otherwise be unavailable if using thick provisioning, possibly reducing costs and administrative overhead.

**Thick Provisioned Lazy Zeroed**

**Advantages:**
- Faster to provision than Thick Provisioned Eager Zeroed
- Better performance than Thin Provisioned

**Disadvantages:**
- Slightly slower to provision than Thin Provisioned
- Slower performance than Thick Provisioned Eager Zero
- Does not support clustering features

When vSphere creates a thick provisioned lazy zeroed disk, it allocates the maximum size of the disk to the VMDK, but does nothing else. At the initial access to each block, vSphere first zeros out the block, then writes the data. Performance of a thick provisioned lazy zeroed disk is not as good a thick provisioned eager zero disk because of this added overhead.

**Thick Provisioned Eager Zeroed**

**Advantages:**
- Best performance
- Overwriting allocated disk space with zeros reduces possible security risks
- Supports clustering features such as Microsoft Cluster Server (MSCS) and VMware Fault Tolerance
Disadvantages:

- Longest time to provision

When vSphere creates a thick provisioned eager zeroed disk, it allocates the maximum size of the disk to the VMDK, then zeros out all of that space.

Example: If you create an 80 GB thick provisioned eager zeroed VMDK, vSphere allocates 80 GB and writes 80 GB of zeros.

By overwriting all data in the allocated space with zeros, thick provisioned eager zeroed eliminates the possibility of reading any residual data from the disk, thereby reducing possible security risks.

Thick provisioned eager zeroed VMDKs have the best performance. When a write operation occurs to a thick provisioned eager zeroed disk, vSphere writes to the disk, with none of the additional overhead required by thin provisioned or thick provisioned lazy zeroed formats.
Using the Cisco Nexus 1000v Switch with Ops Manager

Refer to the procedure in this topic to use Ops Manager with the Cisco Nexus 1000v Switch. First, configure Ops Manager through Step 4 in Configuring BOSH Director on vSphere. Then configure your network according to the following steps.

1. From your Pivotal Cloud Foundry (PCF) Ops Manager Installation Dashboard, click the BOSH Director tile.

2. Select Create Networks.

3. Click the network name to configure the network settings. This is default if you have not changed the name.

4. Find the folder name and port group name for the switch, as you configured them in vCenter. For the example vSphere environment pictured below, a user might want to use the switch configured on the beer-apple port group, which is in the drinks-dc folder.
5. In the **vSphere Network Name** field, instead of entering your network name, enter the folder name and port group name for the switch, as you configured them in vCenter. For the example vSphere environment pictured above, you would enter `drinks-dc/beer-apple` to use the switch configured on the `beer-apple` port group.
6. Click Save.

7. Return to Configuring BOSH Director on vSphere to complete the Ops Manager installation.
Using BOSH Resurrector and vSphere HA

This topic describes the BOSH Resurrector and vSphere High Availability (HA), as well as how to enable them for your Pivotal Cloud Foundry (PCF) deployment.

Pivotal recommends BOSH Resurrector, vSphere HA, and a highly-available storage solution to protect the VMs in your deployment.

BOSH Resurrector

The following sections describe the BOSH Resurrector and how to enable it.

About BOSH Resurrector

The BOSH Resurrector increases Pivotal Application Service (PAS) availability in the following ways:

- Reacts to hardware failure and network disruptions by recreating virtual machines on active, stable hosts
- Detects operating system failures by continuously monitoring virtual machines and recreating them as required
- Continuously monitors the BOSH Agent running on each virtual machine and recreates the VMs as required

The BOSH Resurrector continuously monitors the status of all virtual machines in a PAS deployment. The Resurrector also monitors the BOSH Agent on each VM. If either the VM or the BOSH Agent fail, the Resurrector recreates the virtual machine on another active host.

BOSH Resurrector Limitations

The following limitations apply to using the BOSH Resurrector:

- The Resurrector does not monitor or protect the Ops Manager VM or the BOSH Director VM.
- The Resurrector might not be able to resolve issues caused by the loss of an entire host.
- The Resurrector does not monitor or protect data storage.

To mitigate these limitations, Pivotal recommends vSphere HA. For more information, see vSphere HA below.

Enable BOSH Resurrector

To enable the BOSH Resurrector:

1. Log into the Ops Manager web interface.
2. On the Product Dashboard, select BOSH Director.
3. In the left navigation menu, select Director Config.
4. Select Enable VM Resurrector Plugin and click Save.

vSphere HA

The following sections describe vSphere HA and how to enable it.

About vSphere HA

In the event of a host failure, vSphere HA restarts but does not recreate VMs on alternate ESXi hosts in the same cluster. A host failure could result from a sudden power off, crash due to hardware fault, or the vSphere host becoming disconnected from the network. BOSH Resurrector cannot recover from a host failure.

For more information, see vSphere High Availability in the BOSH documentation.

Enable vSphere HA

Follow the steps below to enable vSphere High Availability:

1. Launch the vSphere Management Console.
2. Right-click the cluster that contains the Pivotal Cloud Foundry deployment and select Edit Settings.
3. Select the Turn on vSphere High Availability checkbox.
4. Click **OK** to enable vSphere High Availability on the cluster.
Configuring Pivotal Cloud Foundry SSL Termination for vSphere Deployments

Page last updated:

To use SSL termination in Pivotal Cloud Foundry (PCF), you must configure the Pivotal-deployed HAProxy load balancer or your own load balancer. Pivotal recommends that you use HAProxy in lab and test environments only. Production environments should instead use a highly-available customer-provided load balancing solution.

Select an SSL termination method to determine the steps you must take to configure Pivotal Application Service (PAS).

Using the Pivotal HAProxy Load Balancer

PCF deploys with a single instance of HAProxy for use in lab and test environments. You can use this HAProxy instance for SSL termination and load balancing to the PCF Routers. HAProxy can generate a self-signed certificate if you do not want to obtain a signed certificate from a well-known certificate authority.

Note: Certificates generated in PAS are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as "Self-Signed Certificates" in the Ops Manager GUI and throughout this documentation.

To use the HAProxy load balancer, you must create a wildcard A record in your DNS and configure three fields in the PAS product tile.

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the System Domain and Apps Domain that you configure in the Domains section of the PAS tile with the HAProxy IP address. For example, with cf.example.com as the main subdomain for your CF install and an HAProxy IP address 203.0.113.1, you must create an A record in your DNS that serves example.com and points *.cf to 203.0.113.1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.cf</td>
<td>A</td>
<td>203.0.113.1</td>
<td>example.com</td>
</tr>
</tbody>
</table>

2. Use the Linux `host` command to test your DNS entry. The `host` command should return your HAProxy IP address.

Example:

$ host cf.example.com
cf.example.com has address 203.0.113.1
$ host anything.example.com
anything.cf.example.com has address 203.0.113.1

3. From the PCF Ops Manager Dashboard, click on the PAS tile.

4. Select Networking.

5. Leave the Router IPs field blank. HAProxy assigns the router IPs internally.

6. Enter the IP address for HAProxy in the HAProxy IPs field.

7. Provide your SSL certificate in the SSL Termination Certificate and Private Key field. See Providing a Certificate for your SSL Termination Point for details.

Using Another Load Balancer

Production environments should use a highly-available customer-provided load balancing solution that does the following:

- Provides SSL termination with wildcard DNS location
- Provides load balancing to each of the PCF Router IPs
• Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers

You must register static IP addresses for PCF with your load balancer and configure three fields in the PAS product tile.

1. Register one or more static IP address for PCF with your load balancer.

2. Create an A record in your DNS that points to your load balancer IP address. The A record associates the System Domain and Apps Domain that you configure in the Domains section of the PAS tile with the IP address of your load balancer.

   For example, with `cf.example.com` as the main subdomain for your CF install and a load balancer IP address 198.51.100.1, you must create an A record in your DNS that serves `example.com` and points `*.cf` to 198.51.100.1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.cf</td>
<td>A</td>
<td>198.51.100.1</td>
<td>example.com</td>
</tr>
</tbody>
</table>

3. From the PCF Ops Manager Dashboard, click on the PAS tile.

4. Select Networking.

5. In the Router IPs field, enter the static IP address for PCF that you have registered with your load balancer.

6. Leave the HAProxy IPs field blank.

7. Provide your SSL certificate in the SSL Termination Certificate and Private Key field. See Providing a Certificate for your SSL Termination Point for details.

   **Note:** When adding or removing PCF routers, you must update your load balancing solution configuration with the appropriate IP addresses.

Return to the Installing Pivotal Cloud Foundry Guide
Availability Zones in vSphere

This topic describes availability zones (AZs) in vSphere and an example of how an operator might scale an app across AZs.

Overview

Pivotal defines an AZ as an operator-assigned, functionally-independent segment of network infrastructure. In cases of partial infrastructure failure, a tile distributes and balances all instances of running apps across remaining AZs. Strategic use of Availability Zones contributes to the fault tolerance and high availability of a tile deployment.

vSphere supports distributing deployments across multiple AZs. For more information, see Step 4: Create Availability Zones in Configuring BOSH Director on vSphere.

It is recommended that customers use three Availability Zones to operate a highly available tile deployment.

Balancing Across AZs During Failure: Example Scenario

An operator scales an app to four instances in a tile environment distributed across three availability zones: A1, A2, and A3. The environment allocates the instances according to the Diego Auction.

For a visual representation of this scenario, see the following diagram:

If A1 experiences a power outage or hardware failure, the two app instances running in A1 terminate while the app instances in zones A2 and A3 continue to run.

For a visual representation of this scenario, see the following diagram:
If A1 remains unavailable, the tile balances new instances of the app across the remaining availability zones.

For a visual representation of this scenario, see the following diagram:
Updating NSX-V Security Group and Load Balancer Information

This topic describes how to update security group and load balancer information for Pivotal Cloud Foundry (PCF) deployments using NSX-V on vSphere. To update this information, you must use the Ops Manager API.

See the Ops Manager API documentation for more information about the API.

See Using Edge Services Gateway on VMware NSX for guidance on how to configure the NSX firewall, load balancing, and NAT/SNAT services for PCF on vSphere installations.

Note: Ops Manager v1.11 supports NSX-V v6.2 and later.

Authenticate

To use the Ops Manager API, you must authenticate and retrieve a token from the Ops Manager User Account and Authentication (UAA) server. For instructions, see the Using Ops Manager API topic.

Update Security Group or Load Balancer Information

To update either NSX security group or load balancer information, you use curl to make a PUT request against the api/v0/staged/products/product_guid/jobs/job_guid/resource_config endpoint.

You must first retrieve the GUID of your PCF deployment, and the GUID of the job whose information you want to update.

Do the following:

1. Retrieve a list of staged products:

```
$ curl 'https://OPS-MAN-FQDN/api/v0/staged/products' \
  -H 'Authorization: Bearer UAA-ACCESS-TOKEN'

[ 
  { 
    "product_version": "1.10.6.0", 
    "guid": "p-bosh-dee11e111e1111ee1e1a", 
    "installation_name": "p-bosh", 
    "type": "p-bosh" 
  }, 
  
  { 
    "type": "cf", 
    "product_version": "1.10.8-build.7", 
    "installation_name": "cf-01222ab1111111aa1a", 
    "guid": "cf-01222ab1111111aa1a" 
  } 
]
```

Record the GUID of the `cf` product. In the above example, the GUID is `cf-01222ab1111111aa1a`.

2. Retrieve a list of jobs for your product:
Record the GUID of the job whose security groups you want to update.

3. You can update either your security group information, load balancer information, or both.

- **Security groups**: To update the security groups for your job, use the following command:

```
$ curl https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs \
  -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
  -X PUT \
  -H "Content-Type: application/json"
```

Replace the placeholder values as follows:

- `INSTANCE-TYPE`: The instance type for the job. For the default instance type, use "automatic".
- `DISK-SIZE`: The disk size for the job. For the default persistent disk size, use "automatic".

**Note:** The `persistent_disk` parameter is required to make this API request. For jobs that do not have persistent disks, you must set the value of the parameter to "automatic".

- `SECURITY-GROUP1`, `SECURITY-GROUP2`: The value of the `nsx_security_groups` parameter is a list of the security groups that you want to set for the job. To clear all security groups for a job, pass an empty list with the `[]` value.

- **Load balancers**: To update the load balancers for your job, use the following command:

```
$ curl https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs/JOB-GUID/resource_config \
  -X PUT \
  -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
  -H "Content-Type: application/json"
```

Replace the placeholder values as follows:

- `INSTANCE-TYPE`: The instance type for the job. For the default instance type, use "automatic".
- `DISK-SIZE`: The disk size for the job. For the default persistent disk size, use "automatic".

**Note:** The `persistent_disk` parameter is required to make this API request. For jobs that do not have persistent disks, you must set the value of the parameter to "automatic".

- `EDGE-NAME`: The name of the NSX Edge.
- `POOL-NAME`: The name of the NSX Edge's server pool.
- `SECURITY-GROUP`: The name of the NSX server pool's target security group.
- `PORT`: The name of the port that the VM service is listening on, such as 5000. You can configure more than one load balancer for a job by using additional hashes in the `nsx_lbs` array.
4. Navigate to [OPS-MAN-FQDN] in a browser and log in to the Ops Manager Installation Dashboard.

5. Click Review Pending Changes, then Apply Changes to redeploy.
Updating NSX-T Load Balancer Server Pool Membership

Page last updated:

This topic describes how to update load balancer server pool membership for Pivotal Cloud Foundry (PCF) deployments using NSX-T on vSphere. To update this information, you must use the Ops Manager API.

See the Ops Manager API documentation for more information about the API.

Note: This procedure requires NSX-T v2.2 or later.

Authenticate

To use the Ops Manager API, you must authenticate and retrieve a token from the Ops Manager User Account and Authentication (UAA) server. For instructions, see the Using Ops Manager API topic.

Update Load Balancer Server Pool Membership

To update load balancer server pool membership, do the following:

1. Create VM Extensions
2. Stage and Associate VM Extensions
3. Apply Changes

Create VM Extensions

To create VM extensions, do the following:

1. Create a file named web.json and paste the following. This is the VM extension for the VMs that serve as the HTTP or HTTPS endpoint of the foundation. The BOSH instance group router serves as the HTTP or HTTPS endpoint.

```
{
  "name": "pas_web_vm_extension",
  "cloud_properties": {
    "nsxt": {
      "lb": {
        "server_pools": [
          {
            "name": "pas-web-pool"
          }
        ]
      }
    }
  }
```

2. Create a file named ssh.json and paste the following. This is the VM extension for the VMs that serve as the SSH endpoint of the foundation. The BOSH instance group diego_brain serves as the SSH endpoint. On Small Footprint PAS the instance group is called control.

```
```
3. Create a file named `tcp.json`. This is the VM extension for the VMs that serve as the endpoint of apps that require custom TCP ports. The BOSH instance group `tcp_router` serves as the custom TCP port endpoint On Small Footprint PAS this instance group is not deployed by default.

```
{
    "name": "pas_tcp_vm_extension",
    "cloud_properties": {
        "nsxt": {
            "lb": {
                "server_pools": [
                    {
                        "name": "pas-tcp-pool"
                    }
                ]
            }
        }
    }
}
```

Stage and Associate VM Extensions

To stage and associate VM extensions with the load balancer, do the following:

1. Run the following command to stage the `web.json` VM extension:

```bash
curl "https://OPS-MAN-FQDN/api/v0/staged/vm_extensions" 
-X POST 
-H "Authorization: Bearer UAA-ACCESS-TOKEN" 
-H "Content-Type: application/json" 
-d "$(cat $VM_EXTENSION_FILE_PATH)"
```

Where:
- `OPS-MAN-FQDN` is the fully qualified domain name of your Ops Manager installation.
- `UAA-ACCESS-TOKEN` is the access token you retrieved when authenticating with the Ops Manager API.
- `VM_EXTENSION_FILE_PATH` is the path to the VM extension file.

2. Repeat the previous step for the `ssh.json` and `tcp.json` VM extensions.

3. Run the following command to retrieve a list of staged products:

```bash
curl 'https://OPS-MAN-FQDN/api/v0/staged/products' 
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

4. Record the GUID of the `cf` product. In the following example output, the GUID is `cf01222a111111111a11`. 

© Copyright Pivotal Software Inc, 2013-2019
5. Retrieve the list of jobs for cf:

```
curl https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs'
-H 'Authorization: Bearer UAA-ACCESS-TOKEN'
```

6. Record the value of guid for the router, tcp_router, and diego_brain jobs. See the following example output:

```
{
"jobs": [
  {
    "name": "router",
    "guid": "router-9c37cf48ae7412f2afd1"
  },
  {
    "name": "tcp_router",
    "guid": "tcp_router-6af18efdd18d198edee9"
  },
  {
    "name": "diego_brain",
    "guid": "diego_brain-b49b0b2aed247302c0e1"
  }
...]
```

7. To associate your job with its appropriate VM extension, run the following command:

```
curl "https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/jobs/JOB-GUID/resource\_config" \
  -X PUT \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer UAA-ACCESS-TOKEN" \
  -d '{"instance_type": {"id": "INSTANCE-TYPE"}, 
    "instances": INSTANCE-COUNT, 
    "additional_vm_extensions": ["VM-EXTENSION-NAME"]}'
```

⚠️ warning: Do not use the resource config fields for nsx_lbs and nsx_security_groups. These fields are only for NSX-V.

Where:
- PRODUCT-GUID is the GUID you recorded for the cf product.
- JOB-GUID is the GUID you recorded for the router job.
- INSTANCE-TYPE is the instance type for the job. For the default instance type, use "automatic".
- INSTANCE-COUNT is the number of instances for the job. The default number for each job is visible in the Resource Config section of the Ops Manager UI.
- VM-EXTENSION-NAME is pas_web_vm_extension.

8. Run the previous command again.

Where:
- JOB-GUID is the GUID you recorded for the tcp_router job.
- VM-EXTENSION-NAME is pas_tcp_vm_extension.

9. Run the previous command again.

Where:
- JOB-GUID is the GUID you recorded for the diego_brain job.
- VM-EXTENSION-NAME is pas_ssh_vm_extension.
Apply Changes

To apply changes, do the following:

1. Log in to the Ops Manager Installation Dashboard.

2. Click Review Pending Changes, then Apply Changes to redeploy.
Configuring PAS

This topic describes how to configure Pivotal Application Service (PAS) as part of deploying Pivotal Cloud Foundry (PCF).

Before beginning this procedure, ensure that you have successfully completed the steps to prepare your environment for PCF and install and configure the BOSH Director.

---

**Note:** If you plan to install the PCF IPsec add-on, you must do so before installing any other tiles. Pivotal recommends installing IPsec immediately after Ops Manager, and before installing the PAS Runtime tile.

---

Add PAS to Ops Manager

1. If you have not already downloaded PAS, log in to Pivotal Network, and click PAS.

2. From the **Releases** drop-down, select the release to install and choose one of the following:
   a. Click PAS to download the PAS file.
   b. Click PCF Small Footprint Runtime to download the Small Footprint Runtime file. For more information, see Getting Started with Small Footprint Runtime.

3. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.

4. Click **Import a Product** to add your tile to Ops Manager. For more information, refer to the Adding and Deleting Products topic.

5. Click the Pivotal Application Service tile in the Installation Dashboard.

---

Assign AZs and Networks

**Note:** For Azure environments, this configuration pane is Assign Networks and does not include AZ configuration.

1. Select **Assign AZ and Networks**.

2. Select the first Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.

3. Select all Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

   **Note:** For production deployments, Pivotal recommends at least three Availability Zones for a highly available installation.

4. From the **Network** dropdown, choose the runtime network that you created when configuring the BOSH Director tile.

5. Click **Save**.

---

Configure Domains

1. Select **Domains**. The following image shows the Domains page.
2. Enter the system and application domains. The **System Domain** defines your target when you push apps to PAS. For example, `system.example.com`. The **Apps Domain** defines where PAS should serve your apps. For example, `apps.example.com`. For additional guidance based on your installation method, see the following table:

<table>
<thead>
<tr>
<th>Installation Method</th>
<th>Guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Enter the domains you created when preparing your environment for PCF.</td>
</tr>
<tr>
<td>Terraform</td>
<td>Enter the values for <code>sys_domain</code> and <code>apps_domain</code> from the Terraform output.</td>
</tr>
</tbody>
</table>

**Note:** Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes.

3. Click **Save**.

### Configure Networking

1. For **Router IPs** and **HAProxy IPs**, see the following guidance:

   - For **AWS**, **Azure**, and **GCP**, leave these fields blank. You do not need to complete these fields when deploying PCF on these infrastructures.
   - For **OpenStack** and **vSphere** the values you enter in the **Router IPs** and **HAProxy IPs** fields depend on whether you are using HAProx in your deployment. Use the table below to determine how to complete these fields.

   **Note:** If you choose to assign specific IP addresses in either the **Router IPs** or **HAProxy IPs** field, ensure that these IP addresses are in the subnet that you configured for PAS in Ops Manager.

<table>
<thead>
<tr>
<th>Using HAProx?</th>
<th>Router IPs Field</th>
<th>HAProxy IPs Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the <strong>Router IPs</strong> field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains that you have configured for your deployment to these IP addresses.</td>
<td>Leave this field blank.</td>
</tr>
<tr>
<td>Yes</td>
<td>Leave this field blank.</td>
<td>1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the <strong>HAProxy IPs</strong> field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains that you have configured for your deployment to these IP addresses.</td>
</tr>
</tbody>
</table>

2. For **SSH Proxy IPs** and **TCP Router IPs**, see the following guidance:

   - For **AWS**, **Azure**, and **GCP**, leave these fields blank. You do not need to complete these fields when deploying PCF on these infrastructures.
   - For **OpenStack** and **vSphere**:

     - (Optional) In **SSH Proxy IPs**, add the IP address for your Diego Brain, which will accept requests to SSH into application containers on port 2222.
     - (Optional) In **TCP Router IPs**, add the IP address(es) you would like assigned to the TCP Routers. You enable this feature at the bottom of this screen.

   **Note:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This disables TCP routing.

3. Under **Certificates and Private Key for HAProx and Router**, you must provide at least one **Certificate and Private Key** name and certificate key pair for HAProx and Gorouter. HAProx and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProx and Gorouter.
The following table indicates which option to choose based on your deployment configuration.

<table>
<thead>
<tr>
<th>Deployment Configuration</th>
<th>TLS Option</th>
<th>Additional Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The Load Balancer is terminating TLS, and Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header</td>
<td>TLS terminated for the first time at infrastructure load balancer (default).</td>
<td>Both HAProxy and the Gorouter forward the XFCC header when included in the request.</td>
</tr>
<tr>
<td>• The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAProxy, and HAProxy instance count is &gt; 0</td>
<td>TLS terminated for the first time at HAProxy.</td>
<td>HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header.</td>
</tr>
<tr>
<td>• The Load Balancer is configured to pass through the TLS handshake via TCP to instances of the Gorouter</td>
<td>TLS terminated for the first time at the Gorouter.</td>
<td>The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for ssh directly to the Diego Brain, consider reducing HAProxy instances to 0.</td>
</tr>
</tbody>
</table>

4. **(Optional)** When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. If you want to configure the Gorouter and HAProxy to trust additional CAs, enter your CA certificates under Certificate Authorities Trusted by Router and HAProxy. All CA certificates should be appended together into a single collection of PEM-encoded entries.

5. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy and Gorouter communications. HAProxy and Gorouter use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Securing Traffic into Cloud Foundry.

6. Configure **Logging of Client IPs in CF Router**. The **Log client IPs** option is set by default. To comply with the General Data Protection Regulation (GDPR), select one of the following options to disable logging of client IP addresses:

   - If your load balancer exposes its own source IP address, disable logging of the X-Forwarded-For HTTP header only.
   - If your load balancer exposes the source IP of the originating client, disable logging of both the source IP address and the X-Forwarded-For HTTP header.

7. Under **Configure support for the X-Forwarded-Client-Cert header**, configure PCF handles x-forwarded-client-cert (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment.

The following table indicates which option to choose based on your deployment configuration.
For a description of the behavior of each configuration option, see Forward Client Certificate to Applications.

8. To configure HAProxy to handle client certificates, select one of the following options in the HAProxy behavior for Client Certificate Validation field:

- **HAProxy does not request client certificates.** This option requires mutual authentication, which makes it incompatible with XFCC option TLS terminated for the first time at HAProxy. HAProxy does not request client certificates, so the client does not provide them and no validation occurs. This is the default configuration.

- **HAProxy requests but does not require client certificates.** The HAProxy requests client certificates in TLS handshakes, validates them when presented, but does not require them.

**warning:** Upon upgrade, PAS will fail to receive requests if your load balancer is configured to present a client certificate in the TLS handshake with HAProxy but HAProxy has not been configured with the certificate authority used to sign it. To mitigate this issue, select HAProxy does not request client certificates in the Networking pane or configure the HAProxy with the appropriate CA.

9. To configure Gorouter behavior for handling client certificates, select one of the following options in the Router behavior for Client Certificate Validation field:

- **Router does not request client certificates.** This option is incompatible with the XFCC configuration options TLS terminated for the first time at HAProxy and TLS terminated for the first time at the Router in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.

- **Router requests but does not require client certificates.** The Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.

- **Router requires client certificates.** The Gorouter validates that the client certificate is signed by a Certificate Authority that the Gorouter trusts. If the Gorouter cannot validate the client certificate, the TLS handshake fails.

**warning:** Requests to the platform will fail upon upgrade if your load balancer is configured with client certificates and the Gorouter does not have the certificate authority. To mitigate this issue, select Router does not request client certificates for Router behavior for Client Certificate Validation in the Networking pane.

10. In the **TLS Cipher Suites for Gorouter** field, review the TLS cipher suites for TLS handshakes between Gorouter and front-end clients such as load balancers or HAProxy. The default value for this field is `DHE-RSA-AES128-GCM-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-GCM-SHA384`.

To modify the default configuration, use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format.

Operators should verify that the ciphers are supported by any clients or front-end components that will initiate TLS handshakes with Gorouter. For a list of TLS ciphers supported by Gorouter, see Securing Traffic into Cloud Foundry.

Verify that every client participating in TLS handshakes with Gorouter has at least one cipher suite in common with Gorouter.

**Note:** Specify cipher suites that are supported by the versions configured in the Minimum version of TLS supported by HAProxy and Router field.

**Note:** AWS Classic Load Balancers do not support PCF’s default cipher suites. See TLS Cipher Suite Support by AWS Load Balancers for information about configuring your AWS load balancers and Gorouter.

11. In the **TLS Cipher Suites for HAProxy** field, review the TLS cipher suites for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. The default value for this field is the following: `DHE-RSA-AES128-GCM-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-GCM-SHA384`.

To modify the default configuration, use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format.

Operators should verify that the ciphers are supported by any clients or front-end components that will initiate TLS handshakes with HAProxy.

Verify that every client participating in TLS handshakes with HAProxy has at least one cipher suite in common with HAProxy.

**Note:** Specify cipher suites that are supported by the versions configured in the Minimum version of TLS supported by HAProxy and Router field.

12. Under **HAProxy forwards requests to Gorouter over TLS**, select **Enable** or **Disable** based on your deployment layout.
- **Enable HAProxy forwarding of requests to Router over TLS.** To enable communication between HAProxy and the Gorouter, do the following:

  1. Leave **Enable** selected.
  2. In the **Certificate Authority for HAProxy Backend** field, specify the Certificate Authority (CA) that signed the certificate you configured in the **Certificate and Private Key for HAProxy and Router** field.

    ![Note](https://example.com/note.png)
    - **Note:** If you used the **Generate RSA Certificate** link to generate a certificate, then the CA to specify is the Ops Manager CA, which you can locate at the `/api/v0/certificateAuthorities` endpoint in the Ops Manager API.

  3. Make sure that Gorouter and HAProxy have TLS cipher suites in common in the **TLS Cipher Suites for Router** and **TLS Cipher Suites for HAProxy** fields.

For more information, see [Terminating SSL/TLS at the Load Balancer and Gorouter](#), [Providing a Certificate for Your SSL/TLS Termination Point](#), and [Using the Ops Manager API](#).

- **Disable HAProxy forwarding of requests to Router over TLS.** To use non-encrypted communication between HAProxy and Gorouter, do the following:

  1. Select **Disable**.
  2. If you are not using HAProxy, set the number of HAProxy job instances to 0 on the **Resource Config** page. See **Disable Unused Resources**.

For more information, see [Terminating SSL/TLS at the Gorouter Only](#) and [Terminating SSL/TLS at the Load Balancer Only](#).

13. If you want to force browsers to use HTTPS when making requests to HAProxy, select **Enable** in the HAProxy support for HSTS field, and complete the following optional configuration steps:

   a. **Enter a Max Age in Seconds** for the HSTS request. By default, the age is set to one year. HAProxy will force HTTPS requests from browsers for the duration of this setting.
   b. Select the **Include Subdomains** checkbox to force browsers to use HTTPS requests for all component subdomains.
   c. Select the **Enable Preload** checkbox to force instances of Google Chrome, Firefox, and Safari that access your HAProxy to refer to their built-in lists of known hosts that require HTTPS, of which HAProxy is one. This ensures that the first contact a browser has with your HAProxy is an HTTPS request, even if the browser has not yet received an HSTS header from HAProxy.

14. If you are not using SSL encryption or if you are using self-signed certificates, select **Disable SSL certificate verification for this environment**.

Selecting this checkbox also disables SSL verification for route services and disables mutual TLS app identity verification.

    ![Note](https://example.com/note.png)
    - **Note:** For production deployments, Pivotal does not recommend disabling SSL certificate verification.

15. **(Optional)** If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on HAProxy and Gorouter** checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

16. **(Optional)** Select the **Disable insecure cookies on the Router** checkbox to set the secure flag for cookies generated by the router.

17. **(Optional)** To disable the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the router** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see [Zipkin Tracing in HTTP Headers](#).

18. **(Optional)** To stop the Router from writing access logs to local disk, deselect the **Enable Router to write access logs locally** checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

19. By default, the PAS routers handle traffic for applications deployed to an isolation segment created by the PCF Isolation Segment tile. To configure the PAS routers to reject requests for applications within isolation segments, select the **Routers reject requests for Isolation Segments** checkbox. Do not enable this option without deploying routers for each isolation segment. See the following topics for more information:

   - **Installing PCF Isolation Segment**
   - **Sharding Routers for Isolation Segments**

20. **(Optional)** By default, Gorouter support for the PROXY protocol is disabled. To enable the PROXY protocol, select **Enable support for PROXY protocol in CF Router**. When enabled, client-side load balancers that terminate TLS but do not support HTTP can pass along information from the originating client. Enabling this option may impact Gorouter performance. For more information about enabling the PROXY protocol in Gorouter, see the [HTTP Header Forwarding](#) section in the [Securing Traffic in Cloud Foundry](#) topic.

21. In the **Choose whether to enable route services** section, choose either **Enable route services** or **Disable route services**. Route services are a class of **marketplace services** that perform content transformation on application requests and responses. See the **Route Services** topic for details. If route services are enabled, Bypass security checks for route service lookup may be selected. Enabling this has potential security concerns, but may be needed for backwards compatibility. See the [Route Service Internal Lookup Considerations](#) section of the Route Services topic for details.

22. **(Optional)** If you want to limit the number of app connections to the backend, enter a value in the **Max Connections Per Backend** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.
To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpSumTop` event metrics emitted for each app request.

If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from `Network Metrics` field. The default value is `500`.

23. Under **Enable Keepalive Connections for Router**, select **Enable** or **Disable**. Keep-alive connections are enabled by default. For more information, see `Keepalive Connections` in **HTTP Routing**.

24. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Router Timeout to Backends** field.

25. (Optional) Use the **Frontend Idle Timeout for Gorouter and HAProxy** field to help prevent connections from your load balancer to Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

In general, set the value higher than your load balancer’s backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

See the following table for specific guidance and exceptions to this rule:

<table>
<thead>
<tr>
<th>IaaS</th>
<th>Guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than <code>60</code>.</td>
</tr>
<tr>
<td>Azure</td>
<td>By default, Azure load balancers times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than <code>240</code> to force the load balancer to send the TCP RST.</td>
</tr>
<tr>
<td>GCP</td>
<td>GCP has a default timeout of 600 seconds. For GCP HTTP load balancers, Pivotal recommends a value greater than <code>600</code>. For GCP TCP load balancers, pivot recommends a value less than <code>600</code> to force the load balancer to send a TCP RST.</td>
</tr>
<tr>
<td>Other</td>
<td>Set the timeout value to be greater than that of the load balancer’s backend idle timeout.</td>
</tr>
</tbody>
</table>

**Note:** Do not set a frontend idle timeout lower than six seconds.

26. (Optional) Increase the value of **Load Balancer Unhealthy Threshold** to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

27. (Optional) Modify the value of **Load Balancer Healthy Threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

28. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the **HTTP Headers to Log** field. For example, to support app developers that deploy Spring apps to PCF, you can enter `Spring-specific HTTP headers`.

29. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for **HAProxy Request Max Buffer Size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

30. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:

- **HAProxy Protected Domains:** Enter a comma-separated list of domains to protect from unknown source requests.
- **HAProxy Trusted CIDRs:** Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the Protected Domains can send traffic to PCF.

31. The **Loggregator Port** defaults to `4443` if left blank. For AWS environments that are not using an Application Load Balancer, enter `4443`.

32. For **Container Network Interface Plugin**, select one of the following:

- **Silk:** This option is the default Container Network Interface (CNI) for PAS.
- **External:** Select this if you are deploying the **VMware NSX-T Container Plug-in for PCF**.

- If you select **External**, follow the instructions in `Deploying PAS with NSX-T Networking` in addition to the PAS configuration instructions in this topic.

⚠️ **warning:** The NSX-T integration only works for fresh installs of PCF. If your PAS is already deployed and running with Silk as its CNI, you cannot change the CNI plugin to NSX-T.
33. If you selected Silk in the previous step, review the following fields:

   a. (Optional) You can change the value in the Applications Network Maximum Transmission Unit (MTU) field. Pivotal recommends setting the MTU value for your application network to 1544. Some configurations, such as networks that use GRE tunnels, may require a smaller MTU value.

   b. (Optional) Enter an IP range for the overlay network in the Overlay Subnet box. If you do not set a custom range, Ops Manager uses 10.255.0.0/16.

   c. Enter a UDP port number in the VXLAN Tunnel Endpoint Port box. If you do not set a custom port, Ops Manager uses 4789.

   d. For Denied logging interval, set the per-second rate limit for packets blocked by either a container-specific networking policy or by the Application Security Group rules applied across the space, org, or deployment. This field defaults to 1.

   e. For UDP logging interval, set the per-second rate limit for UDP packets sent and received. This field defaults to 100.

   f. To enable logging for app traffic, select Log traffic for all accepted/denied application packets. See Manage Logging for Container-to-Container Networking for more information.

   g. The Enable Silk Policy Enforcement checkbox is enabled by default. To disable Silk network policy enforcement between apps, disable the checkbox. Disabling network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

34. For DNS Search Domains, enter DNS search domains for your containers as a comma-separated list. DNS on your containers appends these names to its host names, to resolve them into full domain names.

35. For Database Connection Timeout, set the connection timeout for clients of the policy server and silk databases. The default value is 120. You may need to increase this value if your deployment experiences timeout issues related to Container-to-Container Networking.

36. (Optional) TCP Routing is disabled by default. You should enable this feature if your DNS sends TCP traffic through a load balancer rather than directly to a TCP router. To enable TCP routing:

   a. Select Enable TCP Routing.

   b. For TCP Routing Ports, enter a single port or a range of ports for the load balancer to forward to. These are the same ports that you configured in the Pre-Deployment Steps of the Enabling TCP Routing topic.

      - To support multiple TCP routes, Pivotal recommends allocating multiple ports.
      - To allocate a list of ports rather than a range:

         1. Enter a single port in the TCP Routing Ports field.
         2. After deploying PAS, follow the directions in Configuring a List of TCP Routing Ports to add TCP routing ports using the cf CLI.

   c. For GCP, you also need to specify the name of a GCP TCP load balancer in the LOAD BALANCER column of TCP Router job of the Resource Config screen. You configure this later on in PAS. See the Configure Resources section of this topic.

   d. For AWS, you also need to specify the name of a TCP ELB in the LOAD BALANCER column of TCP Router job of the Resource Config screen. You configure this later on in PAS. For more information, see the Configure Resources section of this topic.

   e. For Azure, you also need to specify the name of Azure load balancer in the LOAD BALANCER column of TCP Router job of the Resource Config screen. You configure this later on in PAS. See the Configure Resources section of this topic.

   f. For OpenStack and vSphere: Return to the top of the Networking screen. In TCP Router IPs field, make sure you have entered IP addresses within your subnet CIDR block. These will be the same IP addresses you configured your load balancer with in Pre-Deployment Steps, unless you configured DNS to resolve the TCP domain name directly to an IP you’ve chosen for the TCP router.

37. (Optional) To disable TCP routing, click Select this option if you prefer to enable TCP Routing at a later time For more information, see the Configuring TCP Routing in PAS topic.

38. (Optional) To enable the creation of dynamic egress policies that allow apps to reach external services, select Enable Beta Dynamic Egress Enforcement. For more information about this feature, see Administering Dynamic Egress Policies (Beta).

39. (Optional) For additional security, enter headers that you want the router to remove from app responses in Remove Specified HTTP Response Headers.

40. Click Save.
Configure Networking - Service Mesh (Beta)

PAS includes an optional, beta routing plane that uses a service mesh. A service mesh provides traffic management, security, and observability for microservices. It enables new features such as weighted routing. For more information, see Service Mesh (Beta).

To add service mesh to your deployment, do the following:

1. Under Service Mesh (Beta), select Enable.

2. For IP Addresses for Ingress Router, do the following depending on your IaaS:
   - vSphere: Enter static IPs for the Istio Routers. You must configure your load balancer with these IPs as well.
   - Other: Leave this field blank.

3. For Ingress Router TLS Keypairs, complete the following fields. You can add more than one keypair if desired using the Add button.
   - Name: Enter a name for the keypair.
   - Certificate and Private Key for Istio Router: Enter the private key and certificate for TLS handshakes with clients. These must be in PEM block format.

4. Click Save.

5. Create a load balancer and add it to the Resource Config pane as described in Deploying Service Mesh (Beta).

Configure Application Containers

1. Select Application Containers. The following image shows the Application Containers page.

2. The Enable Custom Buildpacks checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Disable this option by disabling the checkbox. For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.

3. The Allow SSH access to app containers checkbox controls SSH access to application instances. Enable the checkbox to permit SSH access across your deployment, and disable it to prevent all SSH access. See the Application SSH Overview topic for information about SSH access permissions at the space and app scope.

© Copyright Pivotal Software Inc, 2013-2019
4. If you want to enable SSH access for new apps by default in spaces that allow SSH, select Enable SSH when an app is created. If you deselect the checkbox, developers can still enable SSH after pushing their apps by running the `cf enable-ssh APP-NAME` command.

5. To enable the Gorouter to verify app identity using TLS, click Router uses TLS to verify application identity. To enable the Gorouter and your apps to verify each other's identity using TLS, click Router and applications use mutual TLS to verify each other's identity.

   **Note:** Mutual TLS app identity verification does not work for Windows containers.

Verifying app identity using TLS or mutual TLS enables encryption between router and app containers and guards against misrouting during control plane failures. For more information about Gorouter route consistency modes, see Preventing Misrouting in HTTP Routing.

   **Note:** To support mutual TLS app identity verification, you need v2.3 or later of both PAS and PCF Isolation Segment (IST). The Gorouter and cell components in PCF v2.2 and earlier do not support mTLS handshakes.

6. You can configure Pivotal Application Service (PAS) to run app instances in Docker containers by supplying their IP address ranges in the Private Docker Insecure Registry Whitelist textbox. See the Using Docker Registries topic for more information.

7. Select your preference for Docker Images Disk-Cleanup Scheduling on Cell VMs. If you choose Clean up disk space once usage fills disk, enter a reserved amount of disk space for other jobs in megabytes. For more information about the configuration options and how to configure a reserved amount, see Configuring Docker Images Disk-Cleanup Scheduling.

8. Enter a number in the Max Inflight Container Starts textbox. This number configures the maximum number of started instances across the Diego cells in your deployment. For more information about this feature, see Setting a Maximum Number of Started Containers.

9. Under Enabling NFSv3 volume services, select Enable or Disable. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the Enabling NFS Volume Services topic.

   **Note:** In a fresh install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

10. (Optional) To configure LDAP for NFSv3 volume services, do the following:

    - For LDAP Service Account User, enter the username of the service account in LDAP that will manage volume services.
    - For LDAP Service Account Password, enter the password for the service account.
    - For LDAP Server Host, enter the hostname or IP address of the LDAP server.
    - For LDAP Server Port, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
    - For LDAP User Search Base, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.
      For example, a domain named `cloud.example.com` typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`.
    - For LDAP Server CA Cert, you can optionally enter a certificate if your LDAP server supports TLS and you want to enable TLS connections from the NFS driver to your LDAP server. Paste in the root certificate from your CA certificate or your self-signed certificate.

      **Note:** UAA can only parse one certificate entered into this field. If you enter multiple certificates, UAA only uses the first one you entered and ignores the rest. You only need to include one root certificate or self-signed certificate.

11. Select the Format of timestamps in Diego logs, either RFC3339 timestamps or Seconds since the Unix epoch. Fresh PAS v2.2 installations default to RFC3339 timestamps, while upgrades to PAS v2.2 from previous versions default to Seconds since the Unix epoch.

12. You can optionally modify the Default health check timeout. The value configured for this field is the amount of time allowed to elapse between starting up an app and the first healthy response from the app. If the health check does not receive a healthy response within the configured timeout, then the app is declared unhealthy. The default timeout is 60 seconds and the maximum configurable timeout is 600 seconds.

13. Click Save.

Configure Application Developer Controls

1. Select Application Developer Controls. The following image shows the Application Developer Controls page.
2. Enter the **Maximum File Upload Size (MB)**. This is the maximum size of an application upload.

3. Enter the **Default App Memory (MB)**. This is the amount of RAM allocated by default to a newly pushed application if no value is specified with the `cf` CLI.

4. Enter the **Default App Memory Quota per Org**. This is the default memory limit for all applications in an org. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the `cf` CLI to change the default value.

5. Enter the **Maximum Disk Quota per App (MB)**. This is the maximum amount of disk allowed per application.

   **Note:** If you allow developers to push large applications, PAS may have trouble placing them on Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger applications may not successfully re-deploy if there is insufficient disk capacity. Monitor your deployment to ensure your Cells have sufficient disk to run your applications.

6. Enter the **Default Disk Quota per App (MB)**. This is the amount of disk allocated by default to a newly pushed application if no value is specified with the `cf` CLI.

7. Enter the **Default Service Instances Quota per Org**. The specified limit only applies to the first installation of PAS. After the initial installation, operators can use the `cf` CLI to change the default value.

8. Enter the **Staging Timeout (Seconds)**. When you stage an application droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

9. For **Internal Domain**, enter a domain that apps use for internal DNS service discovery. If you specify this domain using `cf push -d`, other PAS apps can reach the pushed app at `APP-NAME.INTERNAL-DOMAIN`. This value defaults to `apps.internal`.
10. Select the **Allow Space Developers to manage network policies** checkbox to permit developers to manage their own network policies for their applications.

11. Click **Save**.

### Configure Application Security Groups

Setting appropriate **Application Security Groups** is critical for a secure deployment. Type `X` in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See [Restricting App Access to Internal PCF Components](#) for instructions.

### Configure Authentication and Enterprise SSO

1. Select **Authentication and Enterprise SSO**. The following image shows the Authentication and Enterprise SSO page.

   ![Authentication and Enterprise SSO page](image)

   Configure your user store access, which can be an internal user store (managed by Cloud Foundry’s UAA) or an external user store (LDAP or SAML). You can also adjust the lifetimes of authentication tokens.

2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server's internal user store, an external SAML identity provider, or an external LDAP server.

   - To use the internal UAA, select the **Internal** option and follow the instructions in the [Configuring UAA Password Policy](#) topic to configure your password policy.
   - To connect to an external identity provider through SAML, scroll down to select the **SAML Identity Provider** option and follow the instructions in the [Configuring PCF for SAML](#) section of the [Configuring Authentication and Enterprise SSO for Pivotal Application Service (PAS)](#) topic.
   - To connect to an external LDAP server, scroll down to select the **LDAP Server** option and follow the instructions in the [Configuring LDAP](#) section of the [Configuring Authentication and Enterprise SSO for PAS](#) topic.

3. Click **Save**.
Configure UAA

1. Select UAA. The following image shows the UUA page.

![Diagram of UUA page]

1. Under Choose the location of your UAA database select one of the following:
   - PAS database (configured on the Databases pane): Use the same database server that other PAS components use. This system database is configured in the Databases pane, and it can be either internal or external.
   - Other external database - Use a separate, dedicated database server for UAA.

   **warning:** Protect whichever database you use in your deployment with a password.

   **Note:** For GCP installations, Pivotal recommends using an external database on Google Cloud SQL.

2. (Optional) If you selected Other external database, complete the fields as follows. Each field includes additional guidance for specific IaaSes and installation methods.
   - For **Hostname**, enter the hostname of the database server.
     - AWS Terraform: Enter the value of `rds_address` in your Terraform output.
     - GCP Terraform: Enter the value of `sql_db_ip` from your Terraform output.
   - For **TCP Port**, enter the port of the database server.
     - AWS Terraform: Enter the value of `rds_port` in the Terraform output.
     - GCP and GCP Terraform: Enter `3306`.
   - For **Username**, specify a unique username that can access this specific database on the database server.
     - AWS Terraform: Enter the value of `rds_username` from your Terraform output.
     - GCP Terraform: Enter the value of `pas_sql_username` from your Terraform output.
   - For **Password**, specify a password for the provided username.
     - AWS Terraform: Enter the value of `rds_password` from your Terraform output.
     - GCP Terraform: Enter the value of `pas_sql_password` from your Terraform output.
   - For **CA Certificate**, enter a certificate to use for encrypting traffic to and from the database.
3. (Optional) Under **JWT Issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.

4. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a certificate. The following domain must be associated with the certificate: `.login.YOUR-SYSTEM-DOMAIN`.

**Note:** The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the `.login.YOUR-SYSTEM-DOMAIN`.

5. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key Password**.

6. (Optional) To override the default value, enter a custom SAML Entity ID in the **SAML Entity ID Override** field. By default, the SAML Entity ID is `http://login.YOUR-SYSTEM-DOMAIN` where `YOUR-SYSTEM-DOMAIN` is set in the **Domains > System Domain** field.

7. For **Signature Algorithm**, choose an algorithm from the dropdown to use for signed requests and assertions. The default value is `SHA256`.

8. (Optional) In the **Apps Manager Access Token Lifetime**, **Apps Manager Refresh Token Lifetime**, **Cloud Foundry CLI Access Token Lifetime**, and **Cloud Foundry CLI Refresh Token Lifetime** fields, change the lifetimes of tokens granted for Apps Manager and Cloud Foundry Command Line Interface (cf) login access and refresh. Most deployments use the defaults.

9. (Optional) In the **Global Login Session Max Timeout** and **Global Login Session Idle Timeout** fields, change the maximum number of seconds before a global login times out. These fields apply to the following:
   - **Default zone sessions**: Sessions in Apps Manager, PCF Metrics, and other web UIs that use the UAA default zones
   - **Identity zone sessions**: Sessions in apps that use a UAA identity zone, such as a Single Sign-On service plan

10. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Customize Username Label (on login page)** and **Customize Password Label (on login page)**.

11. (Optional) The **Proxy IPs Regular Expression** field contains a pipe-delimited set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

12. Click **Save**.

### Configure CredHub

**Note:** Enabling CredHub is not required. However, you cannot leave the fields under **Encryption Keys** blank. If you do not intend to use CredHub, enter any text in the **Name** and **Key** fields as placeholder values.

1. Select **CredHub**.

2. Choose the location of your CredHub database. PAS includes this CredHub database for services to store their service instance credentials. If you choose **PAS database (configured on the Databases pane)**, CredHub uses the same database as other PAS components, whether the database is internal or external.
   - **GCP and GCP Terraform**: Select **PAS database (configured on the Databases pane)** for your CredHub database. If you choose **Other external database**, Runtime CredHub does not work. For more information, see [CredHub Database Cannot be External on GCP](#).

3. If you chose **Other external database**, enter the following:
   - **Hostname**: This is the IP address of your database server.
     - **AWS Terraform**: Enter the value of `rds_address` in your Terraform output.
   - **TCP Port**: This is the port of your database server, such as `3306`.
   - **AWS Terraform**: Enter the value of `rds_port` in the Terraform output.
   - **Username**: This is a unique username that can access your CredHub database on the database server.
4. Under **Encryption Keys**, specify one or more keys to use for encrypting and decrypting the values stored in the CredHub database.

- **Name**. This is the name of the encryption key.
- **Provider**. This is the provider of the encryption key. If you have configured an HSM provider and HSM servers above, select HSM. Otherwise, select **Internal**.
- **Key**. This key is used for encrypting all data. The key must be at least 20 characters long.
  - **Primary**. This checkbox is used for marking the key you specified above as the primary encryption key. You must mark one key as **Primary**. Do not mark more than one key as **Primary**.

5. **(Optional)** To configure CredHub to use an HSM, complete the following fields:

- **HSM Provider Partition**. This is the name of the HSM provider partition.
- **HSM Provider Partition Password**. This password is used to access the HSM provider partition.
- **HSM Provider Client Certificate**. This is the client certificate for the HSM. For more information, see *Create and Register HSM Clients* in the *Preparing CredHub HSMs for Configuration* topic.

6. If your deployment uses any PCF services that support storing service instance credentials in CredHub and you want to enable this feature, select the **Secure Service Instance Credentials** checkbox.

7. Click **Save**.

8. Select the **Resource Config** pane.

9. Under the **Job** column of the **CredHub** row, ensure that the number of instances is set to **2**. This is the minimum instance count required for high availability.

10. Click **Save**.

For more information about using CredHub for securing service instance credentials, see *Securing Service Instance Credentials with Runtime CredHub*.

### Configure System Databases

You can configure PAS to use an internal MySQL database provided with PCF, or you can configure an external database provider for the databases required by PAS.

**Note:** If you are performing an upgrade, do not modify your existing internal database configuration or you may lose data. You must migrate your existing data first before changing the configuration. See *Upgrading Pivotal Cloud Foundry* for additional upgrade information.

### Internal Database Configuration

**Note:** For Runtime CredHub to work on GCP installations, you must use internal MySQL. See *CredHub Database Cannot be External on GCP*.

**Note:** For GCP installations, Pivotal recommends selecting **External** and using Google Cloud SQL. Only use internal MySQL for non-production or
If you want to use internal databases for your deployment, do the following:

1. Select Databases.
2. Select Internal Databases - MySQL - Percona XtraDB Cluster.
3. Click Save.

Then proceed to Configure Internal MySQL to configure high availability for your internal MySQL databases.

External System Database Configuration

Note: If you use external MySQL, you cannot use Runtime CredHub. See CredHub Database Cannot be External on GCP.

1. Ensure that you have a database instance with the following databases created. The steps vary depending on your database type. For an example procedure, see Creating Databases for PAS.

2. In PAS, select Databases.
3. Select the External Databases option.

Note: If you configure external databases, you cannot configure an internal database in the UAA pane.

4. For Hostname, enter the hostname of the database server. If you are installing PCF using Terraform, this value corresponds to the following variable:
   - AWS Terraform: `rds_address`
   - GCP Terraform: `sql_db_ip`

5. For TCP Port, enter the port of the database server.
   - If you are using GCP CloudSQL, enter `3306`.
   - If you are installing PCF on AWS using Terraform, enter the value for `rds_port`.

6. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify a unique username that can access this specific database on the database server and a password for the provided username. If you are installing PCF using Terraform, these values correspond to the following variables:
   - AWS Terraform: `rds_username` and `rds_password`
   - GCP Terraform: `pas_sql_username` and `pas_sql_password`

Note: Ensure that the networkpolicyserver database user has the ALL PRIVILEGES permission.

7. (Optional) If you want to enable TLS for your external databases, paste your Certificate Authority (CA) certificate in the Database CA Certificate field.

Note: TLS is not currently supported for databases that do not include a matching hostname in their server certificate, e.g. GCP. For more information, see Connection Options for External Applications in the GCP documentation.
8. Click **Save**.

**(Optional) Configure Internal MySQL**

**Note:** You only need to configure this section if you have selected one of the **Internal Databases - MySQL** options in the **Databases** section.

1. Select **Internal MySQL**.

2. In the **Replication canary time period** field, leave the default of 30 seconds or modify the value based on the needs of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

3. In the **Replication canary read delay** field, leave the default of 20 seconds or modify the value based on the needs of your deployment. This field configures how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.

4. **(Required):** In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

5. To prohibit the creation of command line history files on the MySQL nodes, disable the **Allow Command History** checkbox.

6. To allow the admin and roadadmin to connect from any remote host, enable the **Allow Remote Admin Access** checkbox. When the checkbox is disabled, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

**Note:** Network configuration and Application Security Groups restrictions may still limit a client’s ability to establish a connection with the databases.

7. For **Cluster Probe Timeout**, enter the maximum amount of time, in seconds, that a new node will search for existing cluster nodes. If left blank, the default value is 10 seconds.

8. For **Max Connections**, enter the maximum number of connections allowed to the database. If left blank, the default value is 1500.

9. If you want to log audit events for internal MySQL, select **Enable server activity logging** under **Server Activity Logging**.

   a. For the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed.

10. Enter values for the following fields:

    - **Load Balancer Healthy Threshold:** Specifies the amount of time, in seconds, to wait until declaring the MySQL Proxy instance started. This allows an external load balancer time to register the instance as healthy.
    - **Load Balancer Unhealthy Threshold:** Specifies the amount of time, in seconds, that the MySQL Proxy continues to accept connections before shutting down. During this period, the Healthcheck reports as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed healthchecks.

11. If you want to enable the MySQL interruptor feature, select the checkbox to **Prevent node auto re-join**. This feature stops all writes to the MySQL database if it notices an inconsistency in the dataset between the nodes. For more information, see the **Interruptor** section in the MySQL for PCF documentation.

12. Click **Save**.

For more information on how to monitor the node health of your MySQL Proxy instances, see [Using the MySQL Proxy](#).

**Configure File Storage**

1. In the PAS tile, select **File Storage**.

2. For **Max Valid Packages per App**, Pivotal recommends leaving the default value of **5**. However, you may want to lower the value if you have strict storage requirements and want to use less disk space.

3. For **Max Staged Droplets per App**, Pivotal recommends leaving the default value of **5**. However, you may want to lower the value if you have strict storage requirements and want to use less disk space.
For Configure your Cloud Controller's filesystem, see Configuring File Storage for PAS.

(Optional) Configure System Logging

You can configure system logging in PAS to forward log messages from PAS component VMs to an external service. Pivotal recommends forwarding logs to an external service for use in troubleshooting.

Note: The following instructions explain how to configure system logging for PAS component VMs. To forward logs from PCF tiles to an external service, you must also configure system logging in each tile. See the documentation for the given tiles for information about configuring system logging.

To configure system logging in PAS, do the following:

1. In the PAS Settings tab, select the System Logging pane. The following image shows the System Logging pane.

2. For Address, enter the IP address of the syslog server.

© Copyright Pivotal Software Inc, 2013-2019
3. For **Port**, enter the port of the syslog server. The default port for a syslog server is **514**.

   Note: The host must be reachable from the PAS network and accept UDP or TCP connections. Ensure the syslog server listens on external interfaces.


5. For **Encrypt syslog using TLS?**, select **Yes** to use TLS encryption when forwarding logs to a remote server.

   a. For **Permitted Peer**, enter either the name or SHA1 fingerprint of the remote peer.

   b. For **TLS CA Certificate**, enter the TLS CA certificate for the remote server.

6. For **Syslog Drain Buffer Size**, enter the number of messages from the Loggregator Agent that the Doppler server can store before it begins to drop messages. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

7. Disable the **Include container metrics in Syslog Drains** checkbox to prevent the CF Drain CLI plugin from including app container metrics in syslog drains. This feature is enabled by default.

8. Enable the **Enable Cloud Controller security event logging** checkbox to include security events in the log stream. This feature logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

9. Enable the **Use TCP for file forwarding local transport** checkbox to transmit logs over TCP. This prevents log truncation, but may cause performance issues.

10. Disable the **Don't Forward Debug Logs** checkbox to forward DEBUG syslog messages to an external service. This checkbox is enabled by default.

   Note: Some PAS components generate a high volume of DEBUG syslog messages. Enabling the **Don't Forward Debug Logs** checkbox prevents PAS components from forwarding the DEBUG syslog messages to external services. However, PAS still writes the messages to the local disk.

11. For **Custom syslog Configuration**, enter a custom syslog rule. For more information about adding custom syslog rules, see Customizing Syslog Rules.

12. Click **Save**.

To configure Ops Manager for system logging, see the Settings section in the Using the Ops Manager Interface topic.

(Optional) Configure Custom Branding and Apps Manager

This section describes how to configure Custom Branding and Apps Manager to customize the appearance and functionality of Apps Manager. For more information about the Custom Branding configuration settings, see Custom Branding Apps Manager.

1. Select **Custom Branding**. The following image shows the Custom Branding page.
2. Click **Save** to save your settings in this section.

3. Select **Apps Manager**. The following image shows the **Configure Apps Manager** page.
4. Select Enable Invitations to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. See the Inviting New Users section of the Managing User Roles with Apps Manager topic for more information.

5. Select Display Marketplace Service Plan Prices to display the prices for your services plans in the Marketplace.
6. Enter the **Supported currencies as JSON** to appear in the Marketplace. Use the format: ```{"CURRENCY-CODE":"SYMBOL"}``. This defaults to ```{"usd":"$","eur":"€"}``.

7. Use **Product Name**, **Marketplace Name**, and **Customize Sidebar Links** to configure page names and sidebar links in the **Apps Manager** and **Marketplace** pages.

8. The **Apps Manager Memory Usage (MB)** field sets the memory limit with which to deploy the Apps Manager app. Use this field to increase the memory limit if the app fails to start with an **out of memory** error.

9. The **Invitations Memory Usage (MB)** field sets the memory limit with which to deploy the Invitations app. Use this field to increase the memory limit if the app fails to start with an **out of memory** error.

10. The **Apps Manager Polling Interval** field provides a temporary fix if Apps Manager usage degrades Cloud Controller response times. In this case, you can use this field to reduce the load on the Cloud Controller and ensure Apps Manager remains available while you troubleshoot the Cloud Controller. Pivotal recommends that you do not keep this field modified as a long term fix because it can degrade Apps Manager performance. You can optionally do the following:
    - Increase the polling interval above the default of 30 seconds.
    - Disable polling by entering 0. This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.

11. The **App Details Polling Interval** field provides an additional way to reduce the load on the Cloud Controller when the **Apps Manager Polling Interval** field is not sufficient. This field controls the rate at which Apps Manager polls for data when a user views the **Overview** page of an app. Pivotal recommends that you do not keep this field modified as a long term fix because it can degrade Apps Manager performance. You can optionally do the following:
    - Increase the polling interval above the default of 10 seconds.
    - Disable polling by entering 0. This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.

12. For **Multi-Foundation Configuration (BETA)**, enter a JSON object to enable multi-foundation support for Apps Manager. Enabling multi-foundation support allows you to manage orgs, spaces, apps, and service instances from multiple PCF foundations from a single Apps Manager interface. For more information, see [Configuring Multi-Foundation Support in Apps Manager](#).

13. Click **Save** to save your settings in this section.

(Optional) Configure Email Notifications

PAS uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Email Notifications** page if you want to enable end-user self-registration.

1. Select **Email Notifications**. The following image shows the **Email Notifications** page.
2. Enter your reply-to and SMTP email information.

   Note: For GCP, you must use port 2525. Ports 25 and 587 are not allowed on GCP Compute Engine.

3. Verify your authentication requirements with your email administrator and use the SMTP Authentication Mechanism dropdown to select None, Plain, or CRAMMD5. If you have no SMTP authentication requirements, select None.

4. If you selected CRAMMD5 as your authentication mechanism, enter a secret in the SMTP CRAMMD5 secret field.

5. Click Save.

   Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI. See Creating and Managing Users with the cf CLI for more information.

(Optional) Configure App Autoscaler

To use App Autoscaler, you must create an instance of the service and bind it to an app. To create an instance of App Autoscaler and bind it to an app, see Set Up App Autoscaler in the Scaling an Application Using App Autoscaler topic.

1. Click App Autoscaler. The following image shows the Configure App Autoscaler page.
2. Review the following settings:

- **Autoscaler Instance Count**: How many instances of the App Autoscaler service you want to deploy. The default value is 3. For high availability, set this number to 3 or higher. You should set the instance count to an odd number to avoid split-brain scenarios during leadership elections. Larger environments may require more instances than the default number.

- **Autoscaler API Instance Count**: How many instances of the App Autoscaler API you want to deploy. The default value is 1. Larger environments may require more instances than the default number.

- **Metric Collection Interval**: How many seconds of data collection you want App Autoscaler to evaluate when making scaling decisions. The minimum interval is 60 seconds, and the maximum interval is 3600 seconds. The default value is 120. Increase this number if the metrics you use in your scaling rules are emitted less frequently than the existing Metric Collection Interval.

- **Scaling Interval**: How frequently App Autoscaler evaluates an app for scaling. The minimum interval is 15 seconds, and the maximum interval is 120 seconds. The default value is 35.

- **Verbose Logging** (checkbox): Enables verbose logging for App Autoscaler. Verbose logging is disabled by default. Select this checkbox to see more detailed logs. Verbose logs show specific reasons why App Autoscaler scaled the app, including information about minimum and maximum instance limits, App Autoscaler’s status. For more information about App Autoscaler logs, see [App Autoscaler Events and Notifications](#).

- **Disable API Connection Pooling**: If you select this option, the Autoscaler API does not reuse HTTP connections. This may be necessary if your Frontend Idle Timeout for Gorouter is set to a low value, such as 1 second.

3. Click **Save**.

### Configure Cloud Controller

1. Click **Cloud Controller**. The following image shows the **Configure the Cloud Controller** page.
2. Enter your **Cloud Controller DB Encryption Key** if all of the following are true:

- You deployed Pivotal Application Service (PAS) previously.
- You then stopped PAS or it crashed.
- You are re-deploying PAS with a backup of your Cloud Controller database.

See [Backing Up Pivotal Cloud Foundry](#) for more information.

3. **CF API Rate Limiting** prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

To disable CF API Rate Limiting, select **Disable** under **Enable CF API Rate Limiting**. To enable CF API Rate Limiting, perform the following steps:

   a. Under **Enable CF API Rate Limiting**, select **Enable**.
   b. For **General Limit**, enter the number of requests a user or client is allowed to make over an hour interval for all endpoints that do not have a custom limit. The default value is **2000**.
   c. For **Unauthenticated Limit**, enter the number of requests an unauthenticated client is allowed to make over an hour interval. The default value is **100**.

4. **(Optional) Enter in seconds your Database Connection Validation Timeout.** By default, the setting is **3600** seconds or 60 minutes. You can enter **-1** to cause Cloud Controller to make an additional query to the database whenever connections are checked out from the pool. Choosing **-1** has performance implications.

5. **(Optional) Enter in seconds your Database Read Timeout.** By default, the setting is **3600** seconds or 60 minutes.

6. Enter **X** to acknowledge that there are no apps running on **cflinuxfs2** in your deployment. If you are upgrading from PCF v2.4 and your deployment still has apps running on **cflinuxfs2**, see [Migrate Apps to cflinuxfs3](#) in Upgrade Preparation Checklist for PCF v2.5. You can proceed with your upgrade once there are no longer any apps running on **cflinuxfs2**. For additional information, see [PAS Only Supports cflinuxfs3](#) in PAS v2.5 Release Notes.

7. Click **Save**.

Configure Smoke Tests
The Smoke Tests errand runs basic functionality tests against your Pivotal Application Service (PAS) deployment after an installation or update. In this section, choose where to run smoke tests. In the **Errands** section, you can choose whether or not to run the Smoke Tests errand.

1. Select **Smoke Tests**. The following image shows the **Smokes Tests** page.

![Smokes Tests page](image)

Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Elastic Runtime deployment domains.

Choose where to deploy applications when running the smoke tests *

- Temporary space within the system organization (This is deleted after smoke tests finish.)
- Specified org and space (The org and space must have a domain available for routing.)

Organization *

Space *

Domain *

**Save**

2. If you have a shared apps domain, select **Temporary space within the system organization**, which creates a temporary space within the system organization for running smoke tests and deletes the space afterwards. Otherwise, select **Specified org and space** and complete the fields to specify where you want to run smoke tests.

3. Click **Save**.

(Optional) Configure Advanced Features

The **Advanced Features** section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

**Diego Cell Memory and Disk Overcommit**

If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want to use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for Diego Cell.

![Note](image)

*Note: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.*

To enable overcommit, do the following:

1. Select **Advanced Features**.

2. Enter the total desired amount of Diego cell memory value in the **Cell Memory Capacity (MB)** field. Refer to the Diego Cell row in the **Resource**
Config tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the Cell Disk Capacity (MB) field. Refer to the Diego Cell row in the Resource Config tab for the current Cell disk capacity settings that this field overrides.

4. Click Save.

Note: Entries made to each of these two fields set the total amount of resources allocated, not the average.

Whitelist for Non-RFC-1918 Private Networks

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other PCF processes.

The Whitelist for non-RFC-1918 Private Networks field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16.

Most PCF deployments do not require any modifications to this field.

To add your private network to the whitelist, do the following:

1. Select Advanced Features.

2. Append a new allow rule to the existing contents of the Whitelist for non-RFC-1918 Private Networks field. Include the word allow, the network CIDR range to allow, and a semi-colon (;) at the end. For example: allow 172.99.0.0/24;

3. Click Save.

CF CLI Connection Timeout

The CF CLI Connection Timeout field allows you to override the default five second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your PCF deployment. This timeout affects the cf CLI command used to push PAS errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in PAS.

To modify the value of the CF CLI Connection Timeout, perform the following steps:

1. Select Advanced Features.

2. Add a value, in seconds, to the CF CLI Connection Timeout field.

3. Click Save.

SMB Volume Services

You can optionally enable SMB volume services so developers can bind existing SMB shares to their apps. For more information, see the Enabling Volume Services topic.

To enable SMB volume services, do the following:

1. Select Enable SMB volume services.

2. Click Save.

3. In the Errands pane, set the SMB Broker Errand to On.

4. Click Save.

Enable TLS for Internal System Database

You can optionally enable TLS for clients of the internal system database. This feature is in beta, and the checkbox is disabled by default. For more information about the internal system database, see Managing Internal Databases.
To enable TLS for clients of the internal system database, do the following:

1. Select **BETA: Enable TLS for internal system database**.
2. Click **Save**.

**Database Connection Limits**

You can configure the maximum number of concurrent database connections that diego and container networking components can have. Use the field beginning with **Maximum number of open connections**... for a given component. The placeholder values for each field are the default values.

When there are not enough connections available, such as during a time of heavy load, components may experience degraded performance and sometimes failure. To resolve or prevent this, you can increase and fine-tune database connection limits for the component.

⚠️ **warning:** Decreasing the value of this field for a component may affect the amount of time it takes for it to respond to requests.

**Rolling App Deployments**

You can optionally disable rolling app deployments. For more information, see the [Rolling App Deployments (Beta)](https://docs.pivotal.io/platform/pivotal-app-service.html) topic.

To disable zero downtime app deployments, do the following:

1. Select **Disable Zero Downtime App Deployments**
2. Click **Save**.

**Optional** Configure Metric Registrar

The PAS tile includes a **Metric Registrar** configuration pane. You can configure the Metric Registrar from this pane in the following ways:

- Enable
- Edit default scraping interval
- Add blacklisted tags

**Enable the Metric Registrar**

The PAS tile does not deploy the Metric Registrar by default.

To enable the Metric Registrar, do the following:

1. In the PAS tile, click **Metric Registrar**.
2. Select the **Enable Metric Registrar** checkbox.

**Edit Default Scraping Interval**

The scraping interval defines how often the Metric Registrar polls custom metric endpoints. The default is 35 seconds.

To edit the Metric Registrar scraping interval, do the following:

1. In the PAS tile, click **Metric Registrar**.
2. Edit the **Endpoint Scraping Interval** field.

Add Blacklisted Tags
To prevent the Metric Registrar from consuming the value of a metric or event tag, you can add the tag to the Blacklisted Tags field. For example, if you tag your metrics with a customer_id, you may want to add customer_id to the list of blacklisted tags. The field accepts a comma-separated list.

By default, the following tags are blacklisted to prevent interference with other products like PCF Metrics that use and rely on such tags.

- deployment
- job
- index
- id

To prevent the Metric Registrar from consuming the value of a metric or event tag, do the following:

1. In the PAS tile, click Metric Registrar.
2. Add the desired tag to the Blacklisted Tags field.

Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of Pivotal Application Service (PAS). There are two types of errands: post-deploy errands run after the product is installed, and pre-delete errands run before the product in uninstalled.

By default, Ops Manager always runs all errands.

The PAS tile Errands pane lets you change these run rules. For each errand, you can select On to run it always or Off to never run it.

For more information about how Ops Manager manages errands, see the Managing Errands in Ops Manager topic.

Note: Several errands, such as App Autoscaler and Notifications, deploy apps that provide services for your deployment. When one of these apps is running, selecting Off for the corresponding errand on a subsequent installation does not stop the app.

- Smoke Test Errand verifies that your deployment can do the following:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces
- Usage Service Errand deploys the Pivotal Usage Service application, which Apps Manager depends on.
- Apps Manager Errand deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the cf CLI. After Apps Manager has been deployed, Pivotal recommends setting this errand to Off for subsequent PAS deployments. For more information about Apps Manager, see the Getting Started with the Apps Manager topic.
- Notifications Errand deploys an API for sending email notifications to your PCF platform users.

Note: The Notifications app requires that you configure SMTP with a username and password, even if you set the value of SMTP Authentication Mechanism to none.

- Notifications UI Errand deploys a dashboard for users to manage notification subscriptions.
- App Autoscaler Errand pushes the App Autoscaler app, which enables you to configure your apps to automatically scale in response to changes in their usage load. See the Scaling an Application Using Autoscaler topic for more information.
- NFS Broker Errand pushes the NFS Broker app, which supports NFS Volume Services for PAS. For more information, see Enabling NFS Volume Services.
- Metric Registrar Smoke Test Errand verifies the Metric Registrar can access custom metrics emitted by an app and convert them into Loggregator metrics.

Note: The Metric Registrar Smoke Test errand runs only if the Metric Registrar is enabled. For more information about enabling and configuring the Metric Registrar, see Configuring the Metric Registrar.

- SMB Broker Application Errand pushes the SMB Broker app, which supports SMB Volume Services for PAS. For more information, see Enabling SMB Volume Services.

Configure Resources
Configure Load Balancing

In the Resource Config pane, you must associate load balancers with the VMs in your deployment to enable traffic. See Configure Load Balancing for PAS.

Note: This step does not apply to vSphere deployments.

(Optional) Scale Down and Disable Resources

Note: The Resource Config pane has fewer VMs if you are installing the Small Footprint Runtime.

Pivotal Application Service (PAS) defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. See the Zero Downtime Deployment and Scaling in CF and the Scaling Instances in PAS topics for more information.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the Resource Config section and using the dropdowns under Instances for each job.

By default, PAS also uses an internal filestore and internal databases. If you configure PAS to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

To disable specific VMs in Ops Manager, do the following:

1. Click Resource Config.
2. If you configured PAS to use an external S3-compatible filestore, enter 0 in Instances in the File Storage field.
3. If you selected External when configuring the UAA, System, and CredHub databases, edit the following fields:
   - MySQL Proxy: Enter 0 in Instances.
   - MySQL Server: Enter 0 in Instances.
   - MySQL Monitor: Enter 0 in Instances.
4. If you disabled TCP routing, enter 0 Instances in the TCP Router field.
5. If you are not using HAProxy, enter 0 Instances in the HAProxy field.
6. Click Save.

Download Stemcell

This step is only required if your Ops Manager does not already have the stemcell version required by PAS. For more information about importing stemcells, see Importing and Managing Stemcells.

1. Open the Stemcell product page in the Pivotal Network. You may have to log in.
2. Download the appropriate stemcell version targeted for your IaaS.
4. Click Import Stemcell to import the downloaded stemcell file.
5. When prompted, enable the Ops Manager product checkbox to stage your stemcell.
6. Click Apply Stemcell to Products.

Complete the PAS Installation
1. Click the Installation Dashboard link to return to the Installation Dashboard.

2. Click Review Pending Changes, then Apply Changes. The install process generally requires a minimum of 90 minutes to complete. The following image shows the Changes Applied window that displays when the installation process successfully completes.
Quick Start PAS Configuration

This topic describes how to minimally configure Pivotal Application Service (PAS) for evaluation or testing purposes. It does not include optional configurations such as external databases or external file storage.

For production deployments, Pivotal recommends following the instructions in the Configuring PAS topic.

Prerequisites

Before beginning this procedure, ensure that you have successfully completed the steps to prepare your environment for PCF and install and configure the BOSH Director.

Add PAS to Ops Manager

To add PAS to Ops Manager, do the following:

1. If you have not already downloaded PAS, log in to Pivotal Network, and click the PAS tile.

2. From the Releases drop-down, select the release to install and choose one of the following:
   a. Click PAS to download the PAS .pivotal file.
   b. Click PCF Small Footprint Runtime to download the Small Footprint Runtime .pivotal file. For more information, see Getting Started with Small Footprint Runtime.

3. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.

4. Click Import a Product to add your tile to Ops Manager. For more information, refer to the Adding and Deleting Products topic.

5. Click the PAS tile in the Installation Dashboard.

Configure PAS

To install PAS with minimal configuration, do the following:

1. Do the procedure in the Assign AZs and Networks section of Configuring PAS.

2. Do the procedure in the Configure Domains section of Configuring PAS.

3. Click the Networking pane.

4. Under Certificates and Private Key for HAProxy and Router, you must provide at least one Certificate and Private Key name and certificate key pair for HAProxy and Gorouter. HAProxy and Gorouter are enabled to receive TLS communication by default. You can configure multiple certificates for HAProxy and Gorouter.

   a. Click the Add button to add a name for the certificate chain and its private key pair. This certificate is the default used by Gorouter and HAProxy. You can either provide a certificate signed by a Certificate Authority (CA) or click on the Generate RSA Certificate link to generate a self-signed certificate in Ops Manager.

   Note: When providing custom certificates, enter them in the following order: wildcard, Intermediate, CA. For more information, see Creating a .pem File for SSL Certificate Installations in the DigiCert documentation.

   Note: If you configured Ops Manager Front End without a certificate, you can use this new certificate to complete Ops Manager configuration. To configure your Ops Manager Front End certificate, see Configure Front End in Preparing to Deploy Ops Manager on GCP Manually.

   Note: Ensure that you add any certificates that you generate in this pane to your infrastructure load balancer.

5. If you are not using SSL encryption or if you are using self-signed certificates, select Disable SSL certificate verification for this environment. Selecting this checkbox also disables SSL verification for route services and disables mutual TLS app identity verification.
6. **Disable HAProxy forwards requests to Router over TLS.** By default, PAS does not deploy HAProxy.

7. Click the **Application Security Groups** pane. Setting appropriate Application Security Groups is critical for a secure deployment. Type *X* in the box to acknowledge that once the Pivotal Application Service (PAS) deployment completes, you will review and set the appropriate application security groups. See [Restricting App Access to Internal PCF Components](#) for instructions.

8. Under **SAML Service Provider Credentials**, enter a certificate and private key to be used by UAA as a SAML Service Provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a self-signed certificate. The following domain must be associated with the certificate: 

   ```
   *.login.YOUR-SYSTEM-DOMAIN
   ```

   **Note:** The Pivotal Single Sign-On Service and Pivotal Spring Cloud Services tiles require the 

   ```
   *.login.YOUR-SYSTEM-DOMAIN
   ```

9. If the private key specified under **Service Provider Credentials** is password-protected, enter the password under **SAML Service Provider Key Password**.

10. In the **CredHub** pane, Under **Encryption Keys**, specify one or more keys to use for encrypting and decrypting the values stored in the CredHub database.

   - **Name.** This is the name of the encryption key.
   - **Provider.** This is the provider of the encryption key. If you have configured an HSM provider and HSM servers above, select HSM. Otherwise, select Internal.
   - **Key.** This key is used for encrypting all data. The key must be at least 20 characters long.

     - **Primary.** This checkbox is used for marking the key you specified above as the primary encryption key. You must mark one key as Primary. Do not mark more than one key as Primary.

11. In the **E-mail address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.

12. In the **Resource Config** pane, you must associate load balancers with the VMs in your deployment to enable traffic. See [Configure Load Balancing for PAS](#).

### Complete the PAS Installation

To complete the PAS installation, do the following:

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Review Pending Changes**, then **Apply Changes**.
Configuring File Storage for PAS

This topic provides instructions for configure file storage for PAS based on your IaaS and installation method. See the section that applies to your use case.

To minimize system downtime, Pivotal recommends using highly resilient and redundant external filestores for your Pivotal Application Service (PAS) file storage. For more factors to consider when selecting file storage, see the Considerations for Selecting File Storage in Pivotal Cloud Foundry topic.

Internal File Storage

Internal file storage is only appropriate for small, non-production deployments.

To use the PCF internal filestore, select Internal WebDAV, and click Save.

AWS

This section describes how to configure file storage for AWS.

Note: If you followed the instructions in Installing PCF on AWS Manually, you created the necessary resources for external S3-compatible file storage.

For production-level PCF deployments on AWS, Pivotal recommends selecting the External S3-Compatible File Store. For more information about production-level PCF deployments on AWS, see the Reference Architecture for Pivotal Cloud Foundry on AWS.

To use an external S3-compatible filestore for PAS file storage, perform the following steps:

1. Select the External S3-Compatible Filestore option and complete the following fields:

- Enter the [https://](https://) URL Endpoint for your region. For example, in the us-west-2 region, enter [https://s3-us-west-2.amazonaws.com/](https://s3-us-west-2.amazonaws.com/).
- If you use an AWS instance profile to manage role information for your filestore, select the S3 AWS with Instance Profile checkbox. For more information, see AWS Identity and Access Management in the AWS documentation.

- Alternatively, enter the Access Key and Secret Key of the [pcf-user](https://) you created when configuring AWS for PCF. If you select the S3 AWS with Instance Profile checkbox and also enter an Access Key and Secret Key, the instance profile will overrule the Access Key and Secret Key.
- From the S3 Signature Version dropdown, select V4 Signature. For more information about S4 signatures, see Signing AWS API Requests in the AWS documentation.
- For Region, enter the region in which your S3 buckets are located. us-west-2 is an example of an acceptable value for this field.
- Select Server-side Encryption to encrypt the contents of your S3 filestore. This option is only available for AWS S3.
- (Optional) If you selected Server-side Encryption, you can also specify a KMS Key ID. PAS uses the KMS key to encrypt files uploaded to the blobstore. If you do not provide a KMS Key ID, PAS uses the default AWS key. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS).
- Enter names for your S3 buckets:

<table>
<thead>
<tr>
<th>Ops Manager Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildpacks</td>
<td>pcf-</td>
<td>This S3 bucket stores app buildpacks.</td>
</tr>
<tr>
<td>Bucket Name</td>
<td>buildpacks-</td>
<td></td>
</tr>
<tr>
<td>Droplets</td>
<td>pcf-</td>
<td>This S3 bucket stores app droplets. Pivotal recommends that you use a unique bucket name for droplets, but you can also use the same name as above.</td>
</tr>
<tr>
<td>Bucket Name</td>
<td>droplets-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bucket</td>
<td></td>
</tr>
</tbody>
</table>
Configure the following depending on whether your S3 buckets have versioning enabled:

- **Versioned S3 buckets:** Enable the **Use versioning for backup and restore** checkbox to archive each bucket to a version.
- **Unversioned S3 buckets:** Disable the **Use versioning for backup and restore** checkbox, and enter a backup bucket name for each active bucket. The backup bucket name must be different from the name of the active bucket it backs up.

For more information about setting up external S3 blobstores, see the **Backup and Restore with External Blobstores** topic in the Cloud Foundry documentation.

2. Click **Save**.

**Note:** For more information regarding AWS S3 Signatures, see the **Authenticating Requests** topic in the AWS documentation.

## Azure

This section describes how to configure file storage for Azure.

For production-level PCF deployments on Azure, the recommended selection is Azure Storage. For more information about production-level PCF deployments on Azure, see the **Reference Architecture for Pivotal Cloud Foundry on Azure**.

For more factors to consider when selecting file storage, see **Considerations for Selecting File Storage in Pivotal Cloud Foundry**.

To use external Azure file storage for your Pivotal Application Service (PAS) filestore, perform the following steps:

1. Select the **External Azure Storage** option.
2. To create a new storage account and storage containers for the PAS filestore, perform the following steps.

- In the Azure Portal, navigate to the **Storage accounts** tab.
- Click on the plus icon to add a new storage account.
- In the **Name** field, enter a unique name (all lowercase, 3 to 24 alphanumeric characters) for the storage account.
- For the **Deployment model**, select **Resource manager**.
- For **Account kind**, select **General purpose**.
- For **Performance**, select **Standard**.
- From the **Replication** dropdown, select **Locally-redundant storage (LRS)**.
- For **Storage service encryption**, select **Disabled**.
- From the **Subscription** dropdown, select the subscription where you want to deploy PCF resources.
- For **Resource group**, select **Use existing** and enter the name of the resource group where you deployed PAS.
- From the **Location** dropdown, select the **Location** where you are deploying PCF.
- Click **Create**.
- After the storage account is created, select the new storage account from the dashboard.
- Navigate to the **Blob Service** section of the storage account, and then click on **Containers** to create one or more containers in this storage account for builds, droplets, resources, and packages.
- In **Blob Service**, select **Soft Delete**. In **Soft Delete**, click **Enabled** to enable soft delete in your Azure storage account.

**Note:** BBR requires that you enable soft delete in your Azure storage account before you enable backup and restore for your Azure blobstores in Ops Manager. You should set a reasonable retention policy to minimize storage costs. For more information on enabling soft delete in your Azure storage account, see the Azure documentation.

- For each container that you create, set the **Access type** to **Private**.
3. In PAS, enter the name of the storage account you created for **Account Name**.

4. In the **Access Key** field, enter one of the access keys provided for the storage account. To obtain a value for this fields, visit the Azure Portal, navigate to the **Storage accounts** tab and click on **Access keys**.

5. For **Environment**, enter the name of the Azure Cloud environment that contains your storage. This value defaults to AzureCloud.

6. For the **Buildpacks Container Name**, enter the container name for storing your app buildpacks.

7. For **Droplets Container Name**, enter the container name for your app droplet storage. Pivotal recommends that you use a unique container name, but you can use the same container name as the previous step.

8. For **Packages Container Name**, enter the container name for packages. Pivotal recommends that you use a unique container name, but you can use the same container name as the previous step.

9. For **Resources Container Name**, enter the container name for resources. Pivotal recommends that you use a unique container name, but you can use the same container name as the previous step.

10. (Optional) To enable backup and restore for your Azure blobstores in PAS, select the **Enable backup and restore (Soft deletes must be enabled for all storage containers listed above)** checkbox.

11. (Optional) To enable PAS to restore your containers to a different Azure storage account than the account where you take backups, do the following:
   a. Under **Restore from Storage Account**, enter the name of the Azure storage account you want to restore your containers from. Leave this field blank if restoring to the same storage account where you take backups.
   b. Under **Restore using Access Key**, enter the access key for the Azure storage account you specified in **Restore from Storage Account**. Leave this field blank if restoring to the same storage account where you take backups.

12. Click **Save**.

**Note:** To enable backup and restore of your PAS tile that uses S3 compatible blobstore, see [Enable External Blobstore Backups](#).

---

**GCP**

This section describes how to configure file storage for GCP. Follow the instructions that correspond to your installation method: Manual or Terraform.

For production-level PCF deployments on GCP, Pivotal recommends selecting **External Google Cloud Storage**. For more information about production-level PCF deployments on GCP, see the [Reference Architecture for Pivotal Cloud Foundry on GCP](#).

**Manual**

This section describes how to configure file storage for GCP if you installed PCF manually.

PAS can use Google Cloud Storage (GCS) as its external filestore by using either a GCP interoperable storage access key or your GCS Service Account. Choose the procedure that corresponds to your use case.

PAS can use Google Cloud Storage (GCS) as its external filestore by using either a GCP inter-operable storage access key or your GCS Service Account. See below for how to configure each option.

**Note:** The Pivotal Application Service for Windows (PASW) tile is incompatible with GCP configured with a GCS file store. If you are deploying PASW in your GCP environment, Pivotal recommends that you select the S3-compatible file store for your environment.

**External Google Cloud Storage with Access Key and Secret Key**

1. Select the **External Google Cloud Storage with Access Key and Secret Key** option
2. Enter values for **Access Key** and **Secret Key**. To obtain the values for these fields:

   a. In the GCP Console, navigate to the **Storage** tab, then click **Settings**.
   b. Click **Interoperability**.
   c. If necessary, click **Enable interoperability access**. If interoperability access is already enabled, confirm that the default project matches the project where you are installing PCF.
d. Click **Create a new key**.

e. Copy and paste the generated values into the corresponding PAS fields. PCF uses these values for authentication when connecting to Google Cloud Storage.

f. Enter the names of the storage buckets you created in **Preparing to Deploy Ops Manager on GCP Manually**:

   - **Buildpacks Bucket Name**: MY-PCF-buildpacks
   - **Droplets Bucket Name**: MY-PCF-droplets
   - **Resources Bucket Name**: MY-PCF-resources
   - **Packages Bucket Name**: MY-PCF-packages

g. Click **Save**.

---

External Google Cloud Storage with Service Account

1. Select the **External Google Cloud Storage with Service Account** option
a. For **GCP Project ID** enter the Project ID on your GCP Console that you want to use for your PAS file storage.
b. For **GCP Service Account Email** enter the email address associated with your GCP account.
c. For **GCP Service Account JSON** enter the account key that you use to access the specified GCP project, in JSON format.
d. Enter the names of the storage buckets you created in *Preparing to Deploy Ops Manager on GCP Manually*:
   - **Buildpacks Bucket Name**: MY-PCF-buildpacks
   - **Droplets Bucket Name**: MY-PCF-droplets
   - **Resources Bucket Name**: MY-PCF-resources
   - **Packages Bucket Name**: MY-PCF-packages

e. Click **Save**.

**Terraform**

This section describes how to configure file storage for GCP if you installed PCF with Terraform.

PAS can use Google Cloud Storage (GCS) as its external filestore by using either a GCP interoperable storage access key or your GCS Service Account. Choose the procedure that corresponds to your use case.

PAS can use Google Cloud Storage (GCS) as its external filestore by using either a GCP interoperable storage access key or your GCS Service Account. See below for how to configure each option.

*Note*: The Pivotal Application Service for Windows (PASW) tile is incompatible with GCP configured with a GCS file store. If you are deploying PASW in your GCP environment, Pivotal recommends that you select the S3-compatible file store for your environment.

**External Google Cloud Storage with Access Key and Secret Key**

1. Select the **External Google Cloud Storage with Access Key and Secret Key** option
2. Enter values for Access Key and Secret Key. To obtain the values for these fields:

   a. In the GCP Console, navigate to the Storage tab, then click Settings.
   b. Click Interoperability.
   c. If necessary, click Enable interoperability access. If interoperability access is already enabled, confirm that the default project matches the project where you are installing PCF.
d. Click Create a new key.
e. Copy and paste the generated values into the corresponding PAS fields. PCF uses these values for authentication when connecting to Google Cloud Storage.
f. Enter the names of the storage buckets you created in Deploying Ops Manager on GCP Using Terraform:

- **Buildpacks Bucket Name**: Enter the value of `buildpacks_bucket` from your Terraform output.
- **Droplets Bucket Name**: Enter the value of `droplets_bucket` from your Terraform output.
- **Resources Bucket Name**: Enter the value of `packages_bucket` from your Terraform output.
- **Packages Bucket Name**: Enter the value of `resources_bucket` from your Terraform output.

g. Click Save.

External Google Cloud Storage with Service Account

1. Select the External Google Cloud Storage with Service Account option
Openstack

For production-level PCF deployments on OpenStack, the recommended selection is **External S3-Compatible**.

For more factors to consider when selecting file storage, see Considerations for Selecting File Storage in Pivotal Cloud Foundry.

To use an external S3-compatible filestore for PAS file storage, perform the following steps:

1. Select the **External S3-Compatible Filestore** option and complete the following fields:
   - Enter the [https://](https://) URL Endpoint for your region. For example, in the us-west-2 region, enter [https://s3-us-west-2.amazonaws.com/](https://s3-us-west-2.amazonaws.com/).
   - If you use an AWS instance profile to manage role information for your filestore, select the **S3 AWS with Instance Profile** checkbox. For more information, see [AWS Identity and Access Management](https://aws.amazon.com/) in the AWS documentation.

   * Alternatively, enter the **Access Key** and **Secret Key** of the `pcf-user` you created when configuring AWS for PCF. If you select the **S3 AWS with Instance Profile** checkbox and also enter an **Access Key** and **Secret Key**, the instance profile will override the Access Key and Secret Key.
From the S3 Signature Version dropdown, select V4 Signature. For more information about S4 signatures, see Signing AWS API Requests in the AWS documentation.

- For Region, enter the region in which your S3 buckets are located. us-west-2 is an example of an acceptable value for this field.

- Select Server-side Encryption to encrypt the contents of your S3 filestore. This option is only available for AWS S3.

- (Optional) If you selected Server-side Encryption, you can also specify a KMS Key ID. PAS uses the KMS key to encrypt files uploaded to the blobstore. If you do not provide a KMS Key ID, PAS uses the default AWS key. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS).

- Enter names for your S3 buckets:

<table>
<thead>
<tr>
<th>Ops Manager Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildpacks Bucket Name</td>
<td>pcf-buildpacks-bucket</td>
<td>This S3 bucket stores app buildpacks.</td>
</tr>
<tr>
<td>Droplets Bucket Name</td>
<td>pcf-droplets-bucket</td>
<td>This S3 bucket stores app droplets. Pivotal recommends that you use a unique bucket name for droplets, but you can also use the same name as above.</td>
</tr>
<tr>
<td>Packages Bucket Name</td>
<td>pcf-packages-bucket</td>
<td>This S3 bucket stores app packages. Pivotal recommends that you use a unique bucket name for packages, but you can also use the same name as above.</td>
</tr>
<tr>
<td>Resources Bucket Name</td>
<td>pcf-resources-bucket</td>
<td>This S3 bucket stores app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.</td>
</tr>
</tbody>
</table>

- Configure the following depending on whether your S3 buckets have versioning enabled:
  - Versioned S3 buckets: Enable the Use versioning for backup and restore checkbox to archive each bucket to a version.
  - Unversioned S3 buckets: Disable the Use versioning for backup and restore checkbox, and enter a backup bucket name for each active bucket. The backup bucket name must be different from the name of the active bucket it backs up.

For more information about setting up external S3 blobstores, see the Backup and Restore with External Blobstores topic in the Cloud Foundry documentation.

2. Click Save.

Note: For more information regarding AWS S3 Signatures, see the Authenticating Requests topic in the AWS documentation.

vSphere

For production-level PCF deployments on vSphere, the recommended selection is External S3-Compatible and the use of an external filestore. For more information about production-level PCF deployments on vSphere, see the Reference Architecture for Pivotal Cloud Foundry on vSphere.

For more factors to consider when selecting file storage, see Considerations for Selecting File Storage in Pivotal Cloud Foundry.
To use an external S3-compatible filestore for PAS file storage, perform the following steps:

1. Select the **External S3-Compatible Filestore** option and complete the following fields:
   - Enter the **https://** URL Endpoint for your region. For example, in the **us-west-2** region, enter **https://s3-us-west-2.amazonaws.com/**.
   - If you use an AWS instance profile to manage role information for your filestore, select the **S3 AWS with Instance Profile** checkbox. For more information, see **AWS Identity and Access Management** in the AWS documentation.
   - Alternatively, enter the **Access Key** and **Secret Key** of the **pcf-user** you created when configuring AWS for PCF. If you select the **S3 AWS with Instance Profile** checkbox and also enter an **Access Key** and **Secret Key**, the instance profile will overrule the Access Key and Secret Key.
   - From the **S3 Signature Version** dropdown, select **V4 Signature**. For more information about S4 signatures, see **Signing AWS API Requests** in the AWS documentation.
   - For **Region**, enter the region in which your S3 buckets are located. **us-west-2** is an example of an acceptable value for this field.
   - Select **Server-side Encryption** to encrypt the contents of your S3 filestore. This option is only available for AWS S3.
   - (Optional) If you selected **Server-side Encryption**, you can also specify a **KMS Key ID**. PAS uses the KMS key to encrypt files uploaded to the blobstore. If you do not provide a KMS Key ID, PAS uses the default AWS key. For more information, see **Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS)**.
   - Enter names for your S3 buckets:

<table>
<thead>
<tr>
<th>Ops Manager Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildpacks Bucket Name</td>
<td>pcf-buildpacks-bucket</td>
<td>This S3 bucket stores app buildpacks.</td>
</tr>
<tr>
<td>Droplets Bucket Name</td>
<td>pcf-droplets-bucket</td>
<td>This S3 bucket stores app droplets. Pivotal recommends that you use a unique bucket name for droplets, but you can also use the same name as above.</td>
</tr>
<tr>
<td>Packages Bucket Name</td>
<td>pcf-packages-bucket</td>
<td>This S3 bucket stores app packages. Pivotal recommends that you use a unique bucket name for packages, but you can also use the same name as above.</td>
</tr>
<tr>
<td>Resources Bucket Name</td>
<td>pcf-resources-bucket</td>
<td>This S3 bucket stores app resources. Pivotal recommends that you use a unique bucket name for app resources, but you can also use the same name as above.</td>
</tr>
</tbody>
</table>

   - Configure the following depending on whether your S3 buckets have versioning enabled:
     - **Versioned S3 buckets**: Enable the **Use versioning for backup and restore** checkbox to archive each bucket to a version.
     - **Unversioned S3 buckets**: Disable the **Use versioning for backup and restore** checkbox, and enter a backup bucket name for each active bucket. The backup bucket name must be different from the name of the active bucket it backs up.
For more information about setting up external S3 blobstores, see the Backup and Restore with External Blobstores topic in the Cloud Foundry documentation.

2. Click Save.

Note: For more information regarding AWS S3 Signatures, see the Authenticating Requests topic in the AWS documentation.
Configuring Load Balancing for PAS

This topic describes how to configure load balancing for Pivotal Application Service (PAS) by entering the names of your load balancers in the Resource Config pane of the PAS tile. This procedure varies by IaaS installation method. See the section below that corresponds to your use case.

AWS

To configure the Gorouter or HAProxy to AWS Elastic Load Balancers, do the following:

1. Record the names of your ELBs. If you followed the procedures in the Installing PCF on AWS Manually topic, you created the following:
   - `pcf-ssh-elb`: A SSH load balancer. This is a Classic Load Balancer.
   - `pcf-tcp-elb`: A TCP load balancer. This is a Classic Load Balancer.
   - `pcf-web-elb`: A web load balancer. This is an Application Load Balancer.
   - `pcf-web-elb-target-group`: a target group for the web load balancer

2. In the PAS tile, click Resource Config.

3. Enter the name of your SSH load balancer depending on which release you are using.
   - PAS: In the Load Balancers field of the Diego Brain row, enter the name of your SSH load balancer: `pcf-ssh-elb`.
   - Small Footprint Runtime: In the Load Balancers field of the Control row, enter the name of your SSH load balancer: `pcf-ssh-elb`.

4. In the Load Balancers field of the Router row, enter the value determined by the type of load balancer you are using:
   - Application Load Balancer: Enter the name of the target group of your web load balancer, prefixed with `alb:`:
     - `alb:pcf-web-elb-target-group`. The prefix indicates to Ops Manager that you entered the name of a target group, and is required for AWS Application Load Balancers or Network Load Balancers.
   - Classic Load Balancer: Enter the name of the load balancer: `pcf-web-elb`.

Note: If you are using HAProxy in your deployment, then put the name of the load balancers in the Load Balancers field of the HAProxy row instead of the Router row. For a high availability configuration, scale up the HAProxy job to more than one instance.
5. In the **Load Balancers** field of the **TCP Router** row, enter the name of your TCP load balancer if you enabled TCP routing: `pcf-tcp-elb`.

**AWS Terraform**

To configure the Gorouter or HAProxy to AWS Network Load Balancers, do the following:

1. In the PAS tile, click **Resource Config**.
2. Enter the name of your SSH load balancer depending on which release you are using.
   - **Pivotal Application Service (PAS):** In the **Load Balancers** field of the **Diego Brain** row, enter the values of `ssh_target_groups` from the Terraform output, prefixed with "alb:".
   - **Small Footprint Runtime:** In the **Load Balancers** field of the **Control** row, enter the values of `ssh_target_groups` from the Terraform output, prefixed with "alb:".
3. In the **Load Balancers** field of the **Router** row, enter all values of `web_target_groups` from the Terraform output, prefixed with "alb:".
   - **Note:** If you are using HAProxy in your deployment, then put the name of the load balancers in the **Load Balancers** field of the **HAProxy** row instead of the **Router** row. For a high availability configuration, scale up the HAProxy job to more than one instance.
4. In the **Load Balancers** field of the **TCP Router** row, enter all values of `tcp_target_groups` from the Terraform output, prefixed with "alb:".
5. Click **Save**.

**Azure**

To configure the Gorouter to Azure Load Balancers, do the following:

1. Select **Resource Config**.
2. Ensure a **Standard** VM type is selected for the **Router** VM. The PAS deployment fails if you select a **Basic** VM type.
3. Retrieve the name(s) of your external ALB by navigating to the Azure portal, clicking **All resources**, and locating your **Load balancer** resource.
   - **Note:** The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource.
4. Locate the **Router** job in the **Resource Config** pane and enter the name of your external ALB in the field under **Load Balancers**.
5. Retrieve the name of your Diego SSH Load Balancer by navigating to the Azure portal, clicking **All resources**, and locating your **Load balancer**.
resource.

6. Locate the Diego Brain job in the Resource Config pane and enter the name of the Diego SSH Load Balancer in the field under Load Balancers.

7. Ensure that the Internet Connected checkboxes are deselected for all jobs.

8. Scale the number of instances as appropriate for your deployment.

Note: For a high availability deployment of PCF on Azure, Pivotal recommends scaling the number of each PAS job to a minimum of three (3) instances. Using three or more instances for each job creates a sufficient number of availability sets and fault domains for your deployment. For more information, see Reference Architecture for Pivotal Cloud Foundry on Azure CF.

Azure Terraform

To configure the Gorouter to Azure Load Balancers, do the following:

1. Select Resource Config.

<table>
<thead>
<tr>
<th>JOB</th>
<th>INSTANCES</th>
<th>PERSISTENT_DISK_TYPE</th>
<th>VM_TYPE</th>
<th>LOAD BALANCER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consul</td>
<td>Automatic 3</td>
<td>Automatic 1 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>NATS</td>
<td>Automatic</td>
<td>None</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>etcd</td>
<td>Automatic 3</td>
<td>Automatic 1 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>File Storage</td>
<td>Automatic 1</td>
<td>Automatic 100 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>Automatic 3</td>
<td>Automatic 100 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>MySQL Server</td>
<td>Automatic 3</td>
<td>Automatic 100 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>Backup Prepare Node</td>
<td>0</td>
<td>Automatic 200 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>UAA</td>
<td>Automatic 2</td>
<td>None</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>Cloud Controller</td>
<td>Automatic 3</td>
<td>Automatic 1 GB</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
<tr>
<td>HAProxy</td>
<td>Automatic 3</td>
<td>None</td>
<td>Standard_Flex (cpu: 1, ram: 2 GB, disk: 1 GB)</td>
<td></td>
</tr>
</tbody>
</table>

   a. Ensure a Standard VM type is selected for the Router VM. The PAS deployment fails if you select a Basic VM type.

2. Enter the value of `web_lb_name` from your Terraform output in the Resource Config pane under Load Balancers for the Router job.

3. Enter the value of `diego_ssh_lb_name` from your Terraform output in the Resource Config pane under Load Balancers for the Diego Brain job.

4. Ensure that the Internet Connected checkboxes are deselected for all jobs.

5. Scale the number of instances as appropriate for your deployment.

Note: For a high availability deployment of PCF on Azure, Pivotal recommends scaling the number of each PAS job to a minimum of three (3) instances. Using three or more instances for each job creates a sufficient number of availability sets and fault domains for your deployment. For more information, see Reference Architecture for Pivotal Cloud Foundry on Azure CF.

GCP

To Configure Gorouter to GCP Load Balancers, do the following:

1. Navigate to the GCP Console and click Load balancing.
You should see the SSH load balancer, the HTTP(S) load balancer, the TCP WebSockets load balancer, and the TCP router that you created in the Preparing to Deploy PCF on GCP topic.

2. Record the name of your SSH load balancer and your TCP WebSockets load balancer, `MY-PCF-ssh-proxy` and `MY-PCF-wss-logs`.

3. Click your HTTP(S) load balancer, `MY-PCF-global-pcf`.

4. Under Backend services, record the name of the backend service of the HTTP(S) load balancer, `MY-PCF-http-lb-backend`.

5. In the PAS tile, click Resource Config.
6. Under the **LOAD BALANCERS** column of the **Router** row, enter a comma-delimited list consisting of the name of your TCP WebSockets load balancer and the name of your HTTP(S) load balancer backend with the protocol prepended. For example, `tcp:MY-PCF-wss-logs,http:MY-PCF-http-lb-backend`.

   **Note:** Do not add a space between key/value pairs in the **LOAD BALANCER** field or it will fail.

   **Note:** If you are using HAProxy in your deployment, then enter the above load balancer values in the **LOAD BALANCERS** field of the **HAProxy** row instead of the **Router** row. For a high availability configuration, scale up the HAProxy job to more than one instance.

7. If you have enabled TCP routing in the **Networking** pane and set up the **TCP Load Balancer in GCP**, add the name of your TCP load balancer, prepended with `tcp:`, to the **LOAD BALANCERS** column of the **TCP Router** row. For example, `tcp:pcf-tcp-router`.

8. Enter the name of your SSH load balancer depending on which release you are using.
   - **PAS:** Under the **LOAD BALANCERS** column of the **Diego Brain** row, enter the name of your SSH load balancer prepended with `tcp:`. For example, `tcp:MY-PCF-ssh-proxy`.
   - **Small Footprint Runtime:** Under the **LOAD BALANCERS** column of the **Control** row, enter the name of your SSH load balancer prepended with `tcp:`.

9. Verify that the **Internet Connected** checkbox for every job is unchecked. When preparing your GCP environment, you provisioned a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses to allow the jobs to reach the Internet.

   **Note:** If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deselect the **Internet Connected** checkboxes. For more information about using NAT in GCP, see the [GCP documentation](https://cloud.google.com/nat).

10. Click **Save**.
GCP Terraform

To Configure Gorouter to GCP Load Balancers, do the following:

1. Click Resource Config.

2. Under the LOAD BALancers column of the Router row, enter a comma-delimited list consisting of the values of ws_router_pool and http_lb_backend_name from your Terraform output. For example, tcp:pcf-cf-ws,http:pcf-httpslb. These are the names of the TCP WebSockets and HTTP(S) load balancers for your deployment.

   **Note:** Do not add a space between key/value pairs in the LOAD BALANCER field or it will fail.

   **Note:** If you are using HAProxy in your deployment, then enter the above load balancer values in the LOAD BALANCERS field of the HAProxy row instead of the Router row. For a high availability configuration, scale up the HAProxy job to more than one instance.

3. If you have enabled TCP routing in the Networking pane, add the value of tcp_router_pool from your Terraform output, prepended with tcp:, to the LOAD BALancers column of the TCP Router row. For example, tcp:pcf-cf-tcp.

4. Enter the name of your SSH load balancer depending on which release you are using.
   - PAS: Under the LOAD BALancers column of the Diego Brain row, enter the value of ssh_router_pool from your Terraform output, prepended with tcp:. For example, tcp:MY-PCF-ssh-proxy.
   - Small Footprint Runtime: Under the LOAD BALancers column of the Control row, enter the value of ssh_router_pool from your Terraform output, prepended with tcp:.

5. Verify that the Internet Connected checkbox for every job is checked. The terraform templates do not provision a Network Address Translation (NAT) box for internet connectivity to your VMs so instead they will be provided with ephemeral public IP addresses to allow the jobs to reach the Internet.

   **Note:** If you want to provision a Network Address Translation (NAT) box to provide internet connectivity to your VMs instead of providing them with public IP addresses, deselect the Internet Connected checkboxes. For more information about using NAT in GCP, see the GCP documentation.

6. Click Save.

Openstack

Unless you are using your own load balancer, you must enable traffic flow to the OpenStack private subnet as follows. Give each HAProxy a way of routing traffic into the private subnet by providing public IP addresses as floating IP addresses.

1. Click Resource Config.
2. Enter one or more IP addresses in **Floating IPs** for each HAProxy.

3. *(Optional)* If you have enabled the TCP Routing feature, enter one or more IP addresses in **Floating IPs** column for each TCP Router.

4. Click **Save**.
Installing PCF Isolation Segment

Page last updated:

This topic describes how to install the PCF Isolation Segment tile, which allows operators to isolate deployment workloads into dedicated resource pools called isolation segments.

Installing the tile installs a single isolation segment. However, you can install multiple isolation segments using the Replicator tool documented in Step 4.

After installing the tile, you must perform the steps in the Create an Isolation Segment section of the Managing Isolation Segments topic to create the isolation segment in the Cloud Controller Database (CCDB). The topic also includes information about managing an isolation segment.

For more information about how isolation segments work, see Isolation Segments.

Step 1: (Optional) Configure Routing

By default, the Pivotal Application Service (PAS) Router handles traffic for your isolation segment. However, you can deploy a dedicated router for your isolation segment instead. For information about configuring and managing routing for isolation segments, see the Routing for Isolation Segments topic.

To deploy a dedicated router, perform the following steps:

1. Add a load balancer in front of the PAS Router. The steps to do this depend on your IaaS, but the setup of the load balancer should mirror the setup of the load balancer for the PAS Router that you configured in the PAS tile.

2. Create a wildcard DNS entry for traffic routed to any app in the isolation segment. For example, *.iso.example.com.

3. Attach the wildcard DNS entry to the load balancer you created.

Step 2: Install the Tile

Perform the following steps to install the PCF Isolation Segment tile:

1. Download the product file from Pivotal Network.

2. Navigate to YOUR-OPSMAN-FQDN in a browser to log in to the Ops Manager Installation Dashboard.

3. Click Import a Product and select the downloaded product file.

4. Under PCF Isolation Segment in the left column, click the plus sign.

Step 3: Configure the Tile

Click the orange PCF Isolation Segment tile to start the configuration process.
Assign AZs and Networks

Perform the following steps to configure the PCF Isolation Segment tile:

1. Click Assign AZs and Networks.

![AZ and Network Assignments](image)

2. Select an availability zone for your singleton jobs, and one or more availability zones to balance other jobs in.

3. Select a network. This network does not need to be the same network where you deployed PAS. For most deployments, operators should create unique networks in which to deploy the Isolation Segment tile. These networks should maintain network reachability with the Diego components so that the cells can reach the Diego Brain and Diego Database VMs.

4. Click Save.

Networking

Perform the following steps to configure the PCF Isolation Segment tile:

1. Click Networking.

![Configure security and routing services for your isolation segment.](image)

2. (Optional): Under Router IPs, enter one or more static IP addresses for the routers that handle this isolation segment. These IP addresses must be within the subnet CIDR block that you defined in the Ops Manager network configuration for your Isolation Segment. If you have a load balancer, configure it to point to these IP addresses.

   ![Note: Entering the static IP addresses is not necessary for deployments running on a public IaaS such as AWS, GCP, or Azure because PCF users specify the IaaS load balancer in the Resource Config section of the PCF Isolation Segment tile.](image)
The following table indicates which option to choose based on your deployment configuration.

<table>
<thead>
<tr>
<th>Deployment Configuration</th>
<th>TLS Option</th>
<th>Additional Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The Load Balancer is terminating TLS, and</td>
<td>TLS terminated for the first time at infrastructure load balancer (default).</td>
<td>Both HAPProxy and the Gorouter forward the XFCC header when included in the request.</td>
</tr>
<tr>
<td>• Load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The Load Balancer is configured to pass through the TLS handshake via TCP to the instances of HAPProxy, and HAPProxy instance count is &gt; 0</td>
<td>TLS terminated for the first time at HAPProxy.</td>
<td>HAPProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header.</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019 494 2.5
Operators should verify that the ciphers are supported by any clients or front-end components that will initiate TLS handshakes with HAProxy.

For a description of the behavior of each configuration option, see [Forward Client Certificate to Applications](https://docs.pivotal.io/gorouter/running/go-server.html#configuring-forwarding-certificates).

9. To configure HAProxy to handle client certificates, select one of the following options in the **HAProxy behavior for Client Certificate Validation** field:

   - **HAProxy does not request client certificates.** This option requires mutual authentication, which makes it incompatible with XFCC option **TLS terminated for the first time at HAProxy**. HAProxy does not request client certificates, so the client does not provide them and no validation occurs. This is the default configuration.

   - **HAProxy requests but does not require client certificates.** The HAProxy requests client certificates in TLS handshakes, validates them when presented, but does not require them.

   **Warning:** Upon upgrade, PAS will fail to receive requests if your load balancer is configured to present a client certificate in the TLS handshake with HAProxy but HAProxy has not been configured with the certificate authority used to sign it. To mitigate this issue, select **HAProxy does not request client certificates** in the Networking pane or configure the HAProxy with the appropriate CA.

10. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Gorouter behavior for Client Certificate Validation** field:

   - **Router does not request client certificates.** This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication. As client certificates are not requested, client will not provide them, and thus validation of client certificates will not occur.

   - **Router requests but does not require client certificates.** The Gorouter requests client certificates in TLS handshakes, validates them when presented, but does not require them. This is the default configuration.

   - **Router requires client certificates.** The Gorouter validates that the client certificate is signed by a Certificate Authority that the Gorouter trusts. If the Gorouter cannot validate the client certificate, the TLS handshake fails.

   **Warning:** Requests to the platform will fail upon upgrade if your load balancer is configured with client certificates and the Gorouter does not have the certificate authority. To mitigate this issue, select **Router does not request client certificates** for **Router behavior for Client Certificate Validation** in the Networking pane.

11. In the **TLS Cipher Suites for Router** field, review the TLS cipher suites for TLS handshakes between Gorouter and front-end clients such as load balancers or HAProxy. The default value for this field is `ECDHE-RSA-AES128-GCM-SHA256:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`.

To modify the default configuration, use an ordered, colon-delimited list of Golang-supported TLS cipher suites in the OpenSSL format.

Operators should verify that the ciphers are supported by any clients or front-end components that will initiate TLS handshakes with Gorouter. For a list of TLS ciphers supported by Gorouter, see [Securing Traffic into Cloud Foundry](https://docs.pivotal.io/gorouter/running/go-server.html#securing-traffic).

Verify that every client participating in TLS handshakes with Gorouter has at least one cipher suite in common with Gorouter.

**Note:** Specify cipher suites that are supported by the versions configured in the Minimum version of TLS supported by HAProxy and Router field.

12. In the **TLS Cipher Suites for HAProxy** field, review the TLS cipher suites for TLS handshakes between HAProxy and its clients such as load balancers and Gorouter. The default value for this field is the following:


To modify the default configuration, use an ordered, colon-delimited list of TLS cipher suites in the OpenSSL format.

Operators should verify that the ciphers are supported by any clients or front-end components that will initiate TLS handshakes with HAProxy.
Verify that every client participating in TLS handshakes with HAProxy has at least one cipher suite in common with HAProxy.

**Note:** Specify cipher suites that are supported by the versions configured in the Minimum version of TLS supported by HAProxy and Router field.

13. Under **HAProxy forwards requests to Router over TLS**, select Enable or Disable based on your deployment layout.
   - **Enable HAProxy forwarding of requests to Router over TLS**. To enable communication between HAProxy and the Gorouter, do the following:
     1. Leave Enable selected.
     2. In the Certificate Authority for HAProxy Backend field, specify the Certificate Authority (CA) that signed the certificate you configured in the Certificate and Private Key for HAProxy and Router field.

   **Note:** If you used the Generate RSA Certificate link to generate a certificate, then the CA to specify is the Ops Manager CA, which you can locate at the /api/v0/certificateAuthorities endpoint in the Ops Manager API.

14. If you are not using SSL encryption or if you are using self-signed certificates, select Disable SSL certificate verification for this environment. Selecting this checkbox also disables SSL verification for route services and disables mutual TLS app identity verification.

   **Note:** For production deployments, Pivotal does not recommend disabling SSL certificate verification.

15. (Optional) If you want HAProxy or the Gorouter to reject any HTTP (non-encrypted) traffic, select the Disable HTTP on HAProxy and Gorouter checkbox. When selected, HAProxy and Gorouter will not listen on port 80.

16. (Optional) Select the Disable insecure cookies on the Router checkbox to set the secure flag for cookies generated by the router.

17. (Optional) To disable the addition of Zipkin tracing headers on the Gorouter, deselect the Enable Zipkin tracing headers on the router checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see Zipkin Tracing in HTTP Headers.

18. (Optional) To stop the Router from writing access logs to local disk, deselect the Enable Router to write access logs locally checkbox. You should consider disabling this checkbox for high traffic deployments since logs may not be rotated fast enough and can fill up the disk.

19. (Optional) By default, Gorouter support for the PROXY protocol is disabled. To enable the PROXY protocol, select Enable support for PROXY protocol in CF Router. When enabled, client-side load balancers that terminate TLS but do not support HTTP can pass along information from the originating client. Enabling this option may impact Gorouter performance. For more information about enabling the PROXY protocol in Gorouter, see the HTTP Header Forwarding sections in the Securing Traffic in Cloud Foundry topic.

20. Bypass security checks for route service lookup has potential security concerns, but may be needed for backwards compatibility. See the Route Service Internal Lookup Considerations section of Route Services for details.

21. (Optional) If you want to limit the number of app connections to the backend, enter a value in the Max Connections Per Backend field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps.

   To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the httpStartStop event metrics emitted for each app request.

   If your deployment uses PCF Metrics, you can also obtain this peak concurrent connection information from Network Metrics. The default value is 500.

22. Under **Enable Keepalive Connections for Router**, select Enable or Disable. Keep-alive connections are enabled by default. For more information, see Keepalive Connections in HTTP Routing.

23. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the Router Timeout to Backends field.
24. (Optional) Increase the value of Load Balancer Unhealthy Threshold to specify the amount of time, in seconds, that the router continues to accept connections before shutting down. During this period, healthchecks may report the router as unhealthy, which causes load balancers to failover to other routers. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

25. (Optional) Modify the value of Load Balancer Healthy Threshold. This field specifies the amount of time, in seconds, to wait until declaring the Router instance started. This allows an external load balancer time to register the Router instance as healthy.

26. (Optional) If app developers in your organization want certain HTTP headers to appear in their app logs with information from the Gorouter, specify them in the HTTP Headers to Log field. For example, to support app developers that deploy Spring apps to PCF, you can enter Spring-specific HTTP headers.

27. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for HAProxy Request Max Buffer Size. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

28. If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
   - HAProxy Protected Domains: Enter a comma-separated list of domains to protect from unknown source requests.
   - HAProxy Trusted CIDRs: Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the Protected Domains can send traffic to PCF.

29. For DNS Search Domains, enter DNS search domains as a comma-separated list. Your containers append these search domains to hostnames to resolve them into full domain names.

30. The Enable Silk Policy Enforcement checkbox is enabled by default. To disable Silk network policy enforcement between apps, disable the checkbox. Disabling network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

31. Select a sharding mode in the Router Sharding Mode field. The options are explained below. For more information, see Sharding Routers for Isolation Segments.
### Application Containers

Perform the following steps to configure the PCF Isolation Segment tile:

1. Click **Application Containers**.

   ![Image](images/er-config-app-vol-svc.png)

   **Private Docker Insecure Registry Whitelist**: Enter one or more private Docker image registries that are secured with self-signed certificates. Use a comma-delimited list in the format `IP:Port` or `Hostname:Port`.  

   **Segment Name**: Enter the name of your isolation segment. This name must be unique across your PCF deployment. Use this name when performing the steps in the [Create an Isolation Segment](../adminguide/isolation-segments.html#create-an-is) section of the Managing Isolation Segments topic to create the isolation segment in the Cloud Controller Database (CCDB).  

   **DNS Servers**: Enter the DNS servers for your isolation segment.  

   **Docker Images Disk-Cleanup Scheduling on Cell VMs**:
   - **Never clean up Cell disk-space**
   - ** Routinely clean up Cell disk-space**
   - **Clean up disk-space once usage fills disk**

   **Reserved disk space for other jobs (MB) (min:1)**

1. (Optional): Under **Private Docker Insecure Registry Whitelist**, enter one or more private Docker image registries that are secured with self-signed certificates. Use a comma-delimited list in the format `IP:Port` or `Hostname:Port`.  

1. Under **Segment Name**, enter the name of your isolation segment. This name must be unique across your PCF deployment. You use this name when performing the steps in the [Create an Isolation Segment](../adminguide/isolation-segments.html#create-an-is) section of the Managing Isolation Segments topic to create the isolation segment in the Cloud Controller Database (CCDB).  

1. Select your preference for **Docker Images Disk-Cleanup Scheduling on Cell VMs**. If you choose **Clean up disk-space once usage fills disk**, enter a **Reserved amount of disk space for other jobs** in megabytes. For more information about the configuration options and how to configure a reserved amount, see [Configuring Docker Images Disk-Cleanup Scheduling](../opsguide/config-cell-cleanup.html).  

1. Under **Enabling NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow application developers to bind existing NFS volumes to their applications for shared file access. For more information, see the [Enabling NFS Volume Services](../opsguide/enable-vol-services.html) topic.

**Note**: In a clean install, NFSv3 volume services are enabled by default. In an upgrade, NFSv3 volume services match the setting of the previous deployment.
For **LDAP Service Account User**, enter the username of the service account in LDAP that will manage volume services.

For **LDAP Service Account Password**, enter the password for the service account.

For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.

For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.

For **LDAP User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

For example, a domain named `cloud.example.com` typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`.

For **LDAP Server CA Cert**, you can optionally enter a certificate if your LDAP server supports TLS and you want to enable TLS connections from the NFS driver to your LDAP server. Paste in the root certificate from your CA certificate or your self-signed certificate.

2. To enable Gorouter to verify app identity using TLS, select the **Router uses TLS to verify application identity** checkbox. Verifying app identity using TLS improves resiliency and consistency for app routes. For more information about Gorouter route consistency modes, see Preventing Misrouting in HTTP Routing.

   **Note:** This feature does not work if the **Disable SSL certificate verification for this environment** checkbox is selected.

3. Click **Save**.

### System Logging

1. In the **System Logging** menu, select an option underneath **Do you want to configure syslog for system components?**. No is selected by default. This setting only affects Diego cell, router, and HAProxy components within the isolation segment, not shared PAS system components.
2. If you want to use syslog, select Yes.

Configure system logging. Complete the External Syslog fields only if using an external syslogd server.

Do you want to configure syslog for system components?*
- [ ] No
- [ ] Yes

Address *

Port *

Transport Protocol *
- [ ] TCP protocol

Enable TLS

Permitted Peer

TLS CA Certificate

- [ ] Use TCP for file forwarding local transport
  Workaround to avoid truncation of very long (>1024 bytes) log lines. May negatively impact performance.

- [ ] Don't Forward Debug Logs

Custom syslog Configuration

Save
3. Enter the address of your external syslog aggregation service in the Address field. The address can be a hostname or IP address.

4. Enter a port number in the Port field.

5. Select a protocol from the Transport Protocol menu. This is the protocol the system uses to transmit logs to syslog.

6. (Optional) Select the Enable TLS option if you want to transmit logs over TLS.

7. Enter a Permitted Peer.

8. Paste the certificate for your TLS certificate authority (CA) in the TLS CA Certificate field.

9. (Optional) Select the Use TCP for file forwarding local transport to transmit logs over TCP, rather than UDP. This prevents log truncation, but may cause performance issues.

10. (Optional) To forward DEBUG syslog messages to an external service, disable the Don’t Forward Debug Logs checkbox. This checkbox is enabled in PCF Isolation Segment by default.

Note: Some PCF Isolation Segment components generate a high volume of DEBUG syslog messages. Using the Don’t Forward Debug Logs checkbox prevents them from being forwarded to external services. PCF Isolation Segment still writes the messages to the local disk.

11. (Optional) To specify a custom syslog rule, enter it in the Custom syslog Configuration field in RainerScript CF syntax. For more information about customizing syslog rules, see Customizing Syslog Rules.

12. Click Save.

Advanced Features

The Advanced Features section of Pivotal Application Service (PAS) includes new functionality that may have certain constraints. Although these features are fully supported, Pivotal recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

If your apps do not use the full allocation of disk space and memory set in the Resource Config tab, you might want to use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the Resource Config settings for Diego Cell.

Note: Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.

To enable overcommit, do the following:

1. Select Advanced Features.

2. Enter the total desired amount of Diego cell memory value in the Cell Memory Capacity (MB) field. Refer to the Diego Cell row in the Resource Config tab for the current Cell memory capacity settings that this field overrides.

3. Enter the total desired amount of Diego cell disk capacity value in the Cell Disk Capacity (MB) field. Refer to the Diego Cell row in the Resource Config tab for the current Cell disk capacity settings that this field overrides.

4. Click Save.

Note: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

SMB Volume Services

You can optionally enable SMB volume services so developers can bind existing SMB shares to their apps. For more information, see the Enabling Volume Services topic.
To enable SMB volume services, do the following:

1. Select **Enable SMB volume services**.
2. Click **Save**.
3. In the **Errands** pane, set the **SMB Broker Errand** to **On**.
4. Click **Save**.

**Configure Router Resources**

1. Select **Resource Config**.

<table>
<thead>
<tr>
<th>JOB</th>
<th>INSTANCES</th>
<th>PERSISTENT STORAGE</th>
<th>VM TYPE</th>
<th>LOAD BALANCERS</th>
<th>INTERNET CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic micro (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diego Cell</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic xlarge disk (cpu: 4, ram: 16 GB, disk: 128 GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HAProxy</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic micro (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. If you are using a dedicated router for your isolation segment, follow the instructions below.

   - **Note**: The configuration settings available in **Resource Config** vary depending on your IaaS.

   - If you have the Load Balancers column in your **Resource Config**:
     - Enter the wildcard DNS entry attached to your load balancer into the **Router** row under **Load Balancers**.
   - If you do not have the Load Balancers column in your **Resource Config**:
     - Navigate to the **Networking** section of the **PCF Isolation Segment** tile.
     - Specify **Router IPs**.
     - Attach the IP addresses to your load balancer manually.

3. If you are not using HAProxy for this isolation segment, set the number of **Instances** to **0**.

**After Tile Configuration**

After you configure the PCF Isolation Segment tile, perform the following steps:

1. Create the isolation segment in the Cloud Controller Database (CCDB) by following the procedure in the **Create an Isolation Segment** section of the **Managing Isolation Segments** topic.
2. Return to the **Ops Manager Installation Dashboard**.
3. Click **Review Pending Changes**, then **Apply Changes** to deploy the tile.

After the tile finishes deploying, see the **Managing Isolation Segments** topic for more information about managing an isolation segment.

**Step 4: (Optional) Copy the Tile**

If you want to create multiple isolation segments, perform the following steps to copy the PCF Isolation Segment tile with the Replicator tool:

1. Download the Replicator tool from the **Pivotal Cloud Foundry Isolation Segment** section of the **Pivotal Network**.
2. Navigate to the directory where you downloaded the Replicator tool.

3. Replicate the tile:

```
./replicator
  --name "YOUR-NAME"
  --path /PATH/TO/ORIGINAL.pivotal
  --output /PATH/TO/COPY.pivotal
```

Replace the values above with the following:

- **YOUR-NAME**: Provide a unique name for the new PCF Isolation Segment tile. The name must be ten characters or less and only contain alphanumeric characters, dashes, underscores, and spaces.
- **/PATH/TO/ORIGINAL**: Provide the absolute path to the original PCF Isolation Segment tile you downloaded from Pivotal Network.
- **/PATH/TO/COPY**: Provide the absolute path for the copy that the Replicator tool produces.

4. Follow the procedures in this topic using the new `.pivotal` file, starting with Step 1.

### Upgrading Replicated Tiles

For information about upgrading replicated Isolation Segment tiles, see [Upgrading Replicated Tiles](#).
Getting Started with Small Footprint PAS

This topic describes the Small Footprint Pivotal Application Services (PAS) tile for Pivotal Cloud Foundry (PCF).

The Small Footprint PAS is a repackaging of the PAS components into a smaller deployment with fewer virtual machines (VMs). The Limitations section describes the limitations that come with a smaller deployment.

Differentiate Small Footprint PAS and PAS

A standard PAS deployment must have at least 14 VMs, but the Small Footprint PAS requires only four.

The following image displays a comparison of the number of VMs deployed by PAS and Small Footprint PAS.

```
<table>
<thead>
<tr>
<th>VMs in PAS</th>
<th>VMs in Small Footprint PAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Controller</td>
<td>Compute</td>
</tr>
<tr>
<td>Cloud Controller Worker</td>
<td>Control</td>
</tr>
<tr>
<td>Consul</td>
<td>File Storage</td>
</tr>
<tr>
<td>Diego BBS</td>
<td>HAProxy</td>
</tr>
<tr>
<td>Diego Brain</td>
<td>Router</td>
</tr>
<tr>
<td>Doppler Server</td>
<td>Backup Restore Node</td>
</tr>
<tr>
<td>Loggregator Traffic Controller</td>
<td>MySQL Monitor</td>
</tr>
<tr>
<td>NATS</td>
<td>TCP Router</td>
</tr>
<tr>
<td>Router</td>
<td>Istio Router</td>
</tr>
<tr>
<td>Syslog Adapter</td>
<td>Istio Control</td>
</tr>
<tr>
<td>Syslog Scheduler</td>
<td>Route Syncer</td>
</tr>
<tr>
<td>UAA</td>
<td>File Storage</td>
</tr>
<tr>
<td>HAProxy</td>
<td>MySQL Monitor</td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>TCP Router</td>
</tr>
<tr>
<td>MySQL Server</td>
<td>Istio Router</td>
</tr>
<tr>
<td>CredHub</td>
<td>Istio Control</td>
</tr>
<tr>
<td>Route Syncer</td>
<td></td>
</tr>
</tbody>
</table>
```

Use Cases

Use the Small Footprint PAS tile for smaller PCF deployments on which you intend to host 2500 or fewer apps, as described in the Limitations section.
you want to use Small Footprint PAS in a production environment, ensure the Limitations described below are not an issue in your use case.

Note: The Small Footprint PAS is compatible with PCF service tiles.

The Small Footprint PAS tile is also ideal for the following use cases:

- **Proof-of-concept installations:**
  - Deploy PCF quickly and with a small footprint for evaluation or testing purposes.

- **Sandbox installations:**
  - Use Small Footprint PAS as a PCF operator sandbox for tasks such as testing compatibility.

- **Service tile R&D:**
  - Test a service tile against Small Footprint PAS instead of a standard PAS deployment to increase efficiency and reduce cost.

**Limitations**

The Small Footprint PAS has the following limitations:

- **Number of app instances:**
  - The tile is not designed to support large numbers of app instances. You cannot scale the number of Compute VMs beyond 10 instances in the Resource Config pane. The Small Footprint PAS is designed to support 2500 or fewer apps.

- **Increasing platform capacity:**
  - You cannot upgrade the Small Footprint PAS to the standard PAS tile. If you expect platform usage to increase beyond the capacity of the Small Footprint PAS, Pivotal recommends using the standard PAS tile.

- **Management plane availability during tile upgrades:**
  - You may not be able to perform management plane operations like deploying new apps and accessing APIs for brief periods during tile upgrades. The management plane is collocated on the Control VM.

- **App availability during tile upgrades:**
  - If you require availability during your upgrades, you must scale your Compute VMs to a highly available configuration. Ensure sufficient capacity exists to move app instances between Compute VM instances during the upgrade.

**Architecture**

You can deploy the Small Footprint PAS tile with a minimum of four VMs, as shown in the image below.

Note: The following image assumes that you are using an external blobstore.
To reduce the number of VMs required for Small Footprint PAS, the Control and Database VMs include colocated jobs that run on a single VM in PAS. See the next sections for details.

For more information about the components mentioned on this page, see the Architecture section of the Cloud Foundry Concepts guide.

Control VM

The Control VM includes the PAS jobs that handle management plane operations, app lifecycles, logging, and user authorization and authentication. Additionally, all errands run on the Control VM, eliminating the need for a VM for each errand and significantly reducing the time it takes to run errands.

The following image shows all the jobs from PAS that are colocated on the Control VM in Small Footprint PAS.

Database VM

The database VM includes the PAS jobs that handle internal storage and messaging.

The following image shows all the jobs from PAS that are colocated on the Database VM in Small Footprint PAS.
Compute VM

The Compute VM is the same as the Diego Cell VM in PAS.

Other VMs (Unchanged)

The following image shows the VMs perform the same functions in both versions of the PAS tile.

Requirements

The following topics list the minimum resources needed to run Small Footprint PAS and Ops Manager on the public IaaSes that PCF supports:

- Installing PCF on AWS
- Installing PCF on Azure
- Installing PCF on GCP
- Installing PCF on vSphere

Installing Small Footprint PAS
To install the Small Footprint PAS tile, follow the instructions for Installing Pivotal Cloud Foundry on your IaaS.

Follow the same installation and configuration steps as for PAS, with the following differences:

- Selecting a product in Pivotal Network:
  - When you navigate to the PAS tile on Pivotal Network, select the Small Footprint release.

- Configuring resources:
  - The Resource Config pane in the Small Footprint PAS tile reflects the differences in VMs discussed in the Architecture section of this topic.
  - Small Footprint PAS does not default to a highly available configuration like PAS does. It defaults to a minimum configuration. To make Small Footprint PAS highly available, scale the VMs to the following instance counts:
    - Compute: 3
    - Control: 2
    - Database: 3
    - Router: 3

- Configuring load balancers:
  - If you are using an SSH load balancer, you must enter its name in the Control VM row of the Resource Config pane. There is no Diego Brain row in Small Footprint Runtime because the Diego Brain is collocated on the Control VM. You can still enter the appropriate load balancers in the Router and TCP Router rows as normal.

Troubleshooting Colocated Jobs Using Logs

If you need to troubleshoot a job that runs on the Control or Database VMs, do the following:

1. Follow the procedures in Advanced Troubleshooting with the BOSH CLI to the log in to the BOSH Director for your deployment:
   - a. Gather Credential and IP Address Information
   - b. SSH into Ops Manager
   - c. Log in to the BOSH Director

2. Use BOSH to list the VMs in your Small Footprint PAS deployment:

   ```bash
   bosh -e MY-ENV -d MY-DEPLOYMENT vms
   ```

   **Note:** If you do not know the name of your deployment, you can run `bosh -e MY-ENV deployments` to list the deployments for your BOSH Director.

See the following example output:
$ bosh -e example-env -d example-deployment vms

Using environment 'example-env' as client 'ops_manager'

Task 182. Done

Deployment 'example-deployment'

<table>
<thead>
<tr>
<th>Instance</th>
<th>Process State</th>
<th>AZ</th>
<th>IPs</th>
<th>VM CID</th>
<th>VM Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup-restore/8d07724c-c7f6-4add-ba69-856f078114f9</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-6e672a47-55b0-47f7-78ad-759f1295183</td>
<td>micro</td>
</tr>
<tr>
<td>compute/81c6947d-477c-4605-9eb6-5d1305a87c0</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-cec14173c-d93e-414c-6830-4ab8c71316c5</td>
<td>xlarge.disk</td>
</tr>
<tr>
<td>compute/2a035435-5048-4664-a26f-76668474</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-e38b50e-5802-4552-a2d055e-5edc4</td>
<td>xlarge.disk</td>
</tr>
<tr>
<td>compute/a8e2a2c4-d26e-42ea-b34f-8cb56e6b125b</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-6d835455-3713-f7a4-47ca-c42d-4d9f559</td>
<td>xlarge.disk</td>
</tr>
<tr>
<td>control/2b1b8027-79fd-433a-8d4-97f5f20d8f6a</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-9760774c-e13c-4483-7866-7b4ab8185d28</td>
<td>xlarge.disk</td>
</tr>
<tr>
<td>ha_proxy/86587c68-45a8-40c2-9a43-52f16ca4b88</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-275618bc-a86d-4c8b-6e2a-6bba96add1</td>
<td>micro</td>
</tr>
<tr>
<td>mysql_monitor/5185d044-e085-4d6f-a6b6-0f6d88ded57</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-e215888-91b3-4433-d4b3-5232dc5ada53</td>
<td>micro</td>
</tr>
<tr>
<td>route/29a0f32a-dcb4-4b02-8735-8f824ca0d80</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-2b7c85d4-592b-476b-6a71-4f94a6c8f93b</td>
<td>micro</td>
</tr>
<tr>
<td>route/725f497c-e7d0-4b13-8932-76dd3537e5</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-6778c0f1-04c9-4edc-7871-5d66d5b098</td>
<td>micro</td>
</tr>
<tr>
<td>router/3427ab-6191-46b8-abd2-c1d8720e942</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.10</td>
<td>vm-3e84353b-19b8-475c-49eb-4b88f67c656</td>
<td>micro</td>
</tr>
<tr>
<td>database/6f1bca5e-863b-4cf1-912f-4514069123f</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-3c8d4f4-cf47-499f-fa9c-9928c4ebf96</td>
<td>large.disk</td>
</tr>
<tr>
<td>tcp_route/81a86f5a-42b-4af8-843-441f1d2e10c4e</td>
<td>running</td>
<td>us-central1-a</td>
<td>10.0.4.10</td>
<td>vm-0ef2bbf-2498f-479e-98b5-086f1896c5a</td>
<td>micro</td>
</tr>
</tbody>
</table>

13 vms

Succeeded

3. Use BOSH to SSH into one of the Small Footprint PAS VMs.

```bash
bosh -e MY-ENV -d MY-DEPLOYMENT ssh VM-NAME/GUID
```

For example, to SSH into the Control VM, run the following:

```bash
$ bosh -e example-env -d example-deployment ssh control/12b1b827-7fd4-433a-9c3-7f6f204d86a
```

4. Run `sudo su` to act as super user.

5. Use `monit` to list the processes running on the VM.

```
monit summary
```

See the following example output that lists the processes running on the Control VM. The processes listed reflect the colocation of jobs as outlined in the Architecture section of this topic.
6. To access logs, navigate to `/var/vcap/sys/log`:

   ```
   cd /var/vcap/sys/log
   ```

7. Run `ls` to list the log directories for each process. See the following example output from the Control VM:

   ```
   adapter cloud_controller_clock file_server nginx_cc route_registrar standard_injector uaa_ctl.err.log
   auctioneer cloud_controller_ng locket nginx_newrelic_plugin routing-api syslog_drain_binder uaa_ctl.log
   bbs cloud_controller_worker loggregator_trafficcontroller myseq syslog_drain_binder syslog_forwarder
   cc_uploader metron_agent policy-server ssh_proxy tps
   cfdot doppler monitor reverse_log_proxy stager uaa
   ```

8. Navigate to the directory of the process that you want to view logs for. For example, for the Cloud Controller process, run `cd cloud_controller_ng/`. From the directory of the process, you can list and view its logs. See the following example output:

   ```
   cloud_controller_ng_ctl.err.log cloud_controller_ng.log.2.gz cloud_controller_ng.log.6.gz cloud_controller_ng_ctl.log
   cloud_controller_ng_ctl.log.1.gz cloud_controller_ng_ctl.log.5.gz cloud_controller_worker_ctl.log
   cloud_controller_ng_ctl.log.3.gz cloud_controller_ng_ctl.log.7.gz cloud_controller_worker_ctl.log
   cloud_controller/ngctlerr.gz cloud_controller/ngctllog.2.gz cloud_controller/ngctllog.6.gz
   cloud_controller/ngctllog.1.gz cloud_controller/ngctllog.5.gz cloud_controller_worker CTL.log
   ```

Release Notes

The Small Footprint PAS tile releases alongside the PAS tile. For more information, see the [PAS Release Notes](#).
Upgrading Pivotal Cloud Foundry

Page last updated:

This topic describes upgrading Pivotal Cloud Foundry (PCF) to v2.5. The upgrade procedure below describes upgrading Ops Manager, Pivotal Application Service (PAS), and product tiles.

Breaking Changes: Read the Release Notes and Breaking Changes for this release, including the Known Issues sections, before starting the upgrade process.

The apps in your deployment continue to run during the upgrade. However, you cannot write to your deployment or make changes to apps during the upgrade.

For details about how upgrading PCF impacts individual PAS components, see What Happens During PAS Upgrades.

Prepare to Upgrade

If you have not already, complete the steps in the Upgrade Preparation Checklist for PCF v2.5.

Upgrade Ops Manager and Installed Products to v2.5

Follow these steps to upgrade Ops Manager and Installed Products to PCF v2.5.

Import Installation to Ops Manager v2.5 VM

1. Download the Ops Manager VM Template v2.5 from the Pivotal Network site.
2. Record the FQDN address of the existing Ops Manager VM.
3. To avoid conflicts, power off the existing Ops Manager VM.
4. Deploy the new Ops Manager VM by following the steps in one of these topics:
   - AWS: Upgrading BOSH Director on AWS
   - Azure: Upgrading BOSH Director on Azure
   - GCP: Upgrading BOSH Director on GCP
   - OpenStack: Deploying Ops Manager to OpenStack
   - vSphere: Deploying Ops Manager on vSphere
5. When redirected to the Welcome to Ops Manager page, select Import Existing Installation.

6. When prompted, enter the Decryption Passphrase for this Ops Manager installation. You set this passphrase during your initial installation of Ops Manager.
7. Click Choose File and browse to the installation ZIP file exported in the Export Your Installation step of your upgrade preparation.

8. Click Import.

9. A Successfully imported installation message appears upon completion.

Import New PAS and Product Tiles

After upgrading to Ops Manager v2.5, upgrade your product versions:

1. Import the product file to your Ops Manager Installation Dashboard.

2. Hover over the product name in Available Products and click Add.

3. Click the newly-added tile to review any configurable options.

4. (Optional) If you use other service tiles, you can upgrade them following the same procedure. See the Upgrading PAS and Other Pivotal Cloud Foundry Products topic for more information.

Perform Your Upgrade

⚠️ warning: If the installation fails or returns errors, contact Support. Do not attempt to roll back the upgrade by restarting the previous (v2.3) Ops Manager VM.

1. Navigate to the Ops Manager Installation Dashboard.

2. Click Review Pending Changes, then Apply Changes. This immediately imports and applies upgrades to all tiles in a single transaction.

3. Click each service tile, select the Status tab, and confirm that all VMs appear and are in good health.

4. After confirming that the new installation functions correctly, remove the Ops Manager v2.3 VM.
Monitor Upgrade

During the upgrade, you can do the following to monitor your PCF foundation and help troubleshoot any issues.

Check Status and Performance

Note/Reason:
Monitor the progress of the upgrade, checking the status of the foundation at various locations.

Product/Component:
PCF

Pivotal recommends live-monitoring your upgrade with PCF Healthwatch, which monitors and alerts on the current health, performance, and capacity of PCF. Healthwatch captures, calculates, stores, visualizes, and alerts on PCF platform metrics, including:

- BOSH-reported VM metrics
- Runtime component metrics
- Key Performance Indicators and Key Scaling Indicators
- Healthwatch-generated super metrics

For more information, see the [PCF Healthwatch documentation](#).

If you are not using PCF Healthwatch, you can do some or all of the following to monitor upgrade progress:

- Run BOSH CLI status checks:
  ```
  bosh -e ALIAS task TASK_NUMBER
  bosh -e ALIAS vms --vitals
  bosh -e ALIAS instances --ps
  ```
- Check app availability
- Run cf CLI Commands
- Check the availability of the Ops Manager UI
- Check NAS performance (if using NAS)
- Check vSphere performance (if on vSphere)

Check Diego State

Note/Reason:
See the [cfdot documentation](#) on GitHub for details.

Product/Component:
PAS

Use the CF Diego Operator Toolkit (cfdot) to check Diego component instance count by current state.

(Optional) Take Snapshots of Storage Metrics

Reason:
Pivotal recommends this if you have a large foundation and have experienced storage issues in the past.

Product/Component:
PCF

Periodically take snapshots of storage metrics.

(Optional) Collect Logs

Reason:
This information helps determine the cause of upgrade issues.

Product/Component:
PCF
If you encounter problems during upgrade, collect the following information:

- All job logs
- Task debug logs for VM upgrade tasks
- Installation log from Ops Manager

**After Upgrade**

After your upgrade, do the following to prepare for use of your new environment, check its health status, and clean up.

**Re-Create BOSH Alias**

Re-create your alias using BOSH:
```
bosh alias-env ALIAS -e DIRECTOR_IP
```

- **Reason:** To log in to BOSH after upgrading PCF, you need to re-create your alias.
- **Product/Component:** BOSH

**Install New cf CLI**

Download the version of the Cloud Foundry Command Line Interface (cf CLI) packaged with the PAS tile on Pivotal Network.

**Check the Health of Your Deployment**

1. Run BOSH CLI commands to check system status:
   - a. `bosh -e ALIAS -d DEPLOYMENT_NAME instances --ps`
   - b. `bosh -e ALIAS vms --vitals`
   - c. `bosh -e ALIAS -d DEPLOYMENT_NAME cck --report`

   - **Note/Reason:** To ensure that all jobs and process are running as expected
   - **Product/Component:** PCF and all tiles

2. Push and horizontally scale a test application.
   - **Reason:** This is a performance test for PAS.
   - **Product/Component:** PCF

3. If you are running PAS MySQL as a cluster, run the `mysql-diag` tool to validate health of the cluster.
   - **Reason:** See the BOSH CLI v2 instructions in the Running mysql-diag topic.
   - **Product/Component:** PAS

**Check Resource Settings**

If you added custom **VM Type** or **Persistent Disk Type** options, ensure that these values are correctly set and were not overwritten.

- **Note/Reason:** Verify that the Ops Manager **Resource Config** pane still lists your custom options.
- **Product/Component:** PCF

**Run BOSH Clean-Up**

Run `bosh -e ALIAS clean-up --all` to clean up old stemcells, releases, orphaned disks, and other unused resources.

- **Product/Component:** Tiles
What Happens During PAS Upgrades

This topic explains what happens to Pivotal Application Service (PAS) components and apps during a PAS upgrade.

BOSH Drains Diego Cell VMs

During a PAS upgrade, BOSH drains all Diego cell VMs that host app instances. BOSH manages this process by upgrading a batch of cells one at a time.

When BOSH triggers an upgrade, each upgrading Diego cell enters evacuation mode. In evacuation mode, BOSH stops Diego cells and then schedules replacements for its app instances.

For more information, see the Specific Guidance for Diego Cells section of the Configuring PAS for Upgrades topic.

cf push Can Become Unavailable

cf push is mostly available for the duration of a PAS upgrade. However, cf push can become unavailable when a single VM is in use or during BOSH Backup and Restore (BBR).

For more information, see cf push Availability During Pivotal Application Service Upgrades.

PAS Components Upgrade

This section describes the order in which Ops Manager upgrades components and runs tasks during a full platform upgrade. It also explains how the scale of different Pivotal Application Service (PAS) components affects uptime during upgrades, and which components are scalable.

When performing an upgrade, Ops Manager first upgrades individual components, and then runs one-time tasks.

1. The Components section describes how Ops Manager upgrades PAS components and explains how individual component upgrades affect broader PAS capabilities.

2. The One-Time Tasks section lists the tasks that Ops Manager runs after it upgrades the PAS components.

Components

Ops Manager upgrades PAS components in a fixed order that honors component dependencies and minimizes downtime and other system limitations during the upgrade process.

The type and duration of downtime and other limitations that you can expect during a PAS upgrade reflect the following:

- Component instance scaling. See How Single-Component Scaling Affects Upgrades
- Component upgrade order. See Component Upgrade Order and Behavior

How Single-Component Scaling Affects Upgrades

In Pivotal Cloud Foundry (PCF) Ops Manager, the Pivotal Application Service (PAS) tile > Resource Config pane shows the components that the BOSH Director installs:

- **Scalable** component fields let you select the instance count from a range of settings or enter a custom value.
- **Unscalable** component fields allow a maximum of one instance.

When a component is scaled at a single instance, it can experience downtime and other limitations while the single VM restarts. This behavior might be acceptable for a test environment. To avoid downtime in a production environment, you should scale any scalable components, such as HAProxy, Router, and Diego cells, to more than one instance.

For more information about how the scale of each component affects upgrade behavior, see the Component Upgrade Order and Behavior table below.
## Component Upgrade Order and Behavior

The table below lists components in the order that Ops Manager upgrades each. It also lists which components are scalable and explains how component downtime affects PAS app and control availability. The table includes the following columns:

- **Scalable**: Indicates whether the component is scalable above a single instance.
- **Extended Downtime**: Indicates that if there is only one instance of the component, that component is unavailable for up to five minutes during an Ops Manager upgrade.
- **Downtime Affects…**: Indicates the plane of the PAS platform that component downtime affects, if the component is scaled at single instance:
  - **Apps**: Downtime can affect app availability.
  - **Platform**: Apps remain available during component downtime, but you cannot push, stage, or restart apps, or run other Cloud Foundry command-line interface (cf CLI) commands.
- **Other Limitations and Information**: Provides the following information:
  - Component availability, behavior, and usage during an upgrade
  - Guidance on disabling the component before an upgrade

### Components and Their Properties

<table>
<thead>
<tr>
<th>Component</th>
<th>Scalable</th>
<th>Extended Downtime</th>
<th>Downtime Affects…</th>
<th>Other Limitations and Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATS</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File Storage</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>The MySQL Proxy is responsible for managing failover of the MySQL Servers. If the Proxy becomes unavailable, then access to the MySQL Server could be broken.</td>
</tr>
<tr>
<td>MySQL Server</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>The MySQL Server is responsible for persisting internal databases for the platform. If the MySQL Server becomes unavailable, then platform services that rely upon a database (Cloud Controller, UAA) will also become unavailable. See <a href="#">Effects of MySQL Downtime</a> for details.</td>
</tr>
<tr>
<td>Backup Restore Node</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diego BBS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>UAA</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>If a user has an active authorization token prior to performing an upgrade, they can still log in using either a UI or the CLI.</td>
</tr>
<tr>
<td>Cloud Controller</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HAProxy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>HAProxy is used to load-balance incoming requests to the Router. If HAProxy is unavailable, you may lose the ability to make requests to applications unless there is another routing path from your load balancer to the Router.</td>
</tr>
<tr>
<td>Router</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>The Router is responsible for routing requests to their application containers. If the Router is not available, then applications cannot receive requests.</td>
</tr>
<tr>
<td>MySQL Monitor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock Global</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloud Controller Worker</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diego Brain</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diego Cell</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>If you only have one Diego Cell, upgrading causes downtime for all apps that run on it. These include apps pushed with <code>cf push</code> and apps automatically installed by PAS, like Apps Manager and the App Usage Service.</td>
</tr>
</tbody>
</table>

**Note:** A full Ops Manager upgrade may take close to two hours, and you will have limited ability to deploy an application during this time.
Loggregator
Trafficcontroller ✓

Ops Manager operators experience 2-5 minute gaps in logging.

Syslog Adapter ✓

Syslog Scheduler ✓

Ops Manager operators experience 2-5 minute gaps in logging.

Doppler Server ✓

TCP Router (if enabled) ✓

CredHub ✓ ✓ ✓ ✓ App downtime for apps that use secure credentials. Platform downtime for cf CLI commands such as `bind-service` and `unbind-service` applied to services configured with CredHub.

Istio Control ✓ ✓ ✓

Istio Router ✓ ✓ ✓ Downtime for this component only affects routes on mesh domains.

Route Syncer ✓ ✓ ✓ Downtime for this component only affects routes on mesh domains.

One-Time Tasks

After Ops Manager upgrades components, it performs system checks and launches UI apps and other PAS components as Cloud Foundry apps. These tasks run in the following order:

1 Apps Manager Errand - Push Apps Manager
2 Smoke Test Errand - Run smoke tests
3 Usage Service Errand - Push Usage Service app
4 Notifications Errand - Push Notifications app
5 Notifications UI Errand - Push Notifications UI
6 App Autoscaler Errand - Push App Autoscaler
7 App Autoscaler Smoke Test Errand - Run smoke tests against App Autoscaler
8 Register Autoscaling Service Broker
9 Destroy Autoscaling Service Broker
10 Bootstrap Errand - Recover MySQL cluster
11 MySQL Rejoin Unsafe Errand

Upgrading Installation Example

For sample performance measurements of an upgrading Cloud Foundry installation, see Upgrade Load Example: Pivotal Web Services ☞.
Upgrade Preparation Checklist for PCF v2.5

This topic serves as a checklist for preparing to upgrade Pivotal Cloud Foundry (PCF) from v2.4 to v2.5.

This topic contains important preparation steps that you must follow before beginning your upgrade. Failure to follow these instructions may jeopardize your existing deployment data and cause the upgrade to fail.

After completing the steps in this topic, you can continue to Upgrading Pivotal Cloud Foundry.

Back Up Your PCF Deployment

Pivotal recommends backing up your PCF deployment before upgrading, to restore in the case of failure. To do this, follow the instructions in the Backing Up Pivotal Cloud Foundry with BBR topic.

Review Changes in PCF v2.5

Review each of the following links to understand the changes in the new release, such as new features, known issues, and breaking changes.

- **Release Notes**
  - [Ops Manager Release v2.5 Release Notes](#)
  - [PAS v2.5 Release Notes](#)

- **Known Issues**
  - [Ops Manager v2.5 Known Issues](#)
  - [PAS v2.5 Known Issues](#)

- **Breaking Changes**
  - [PCF v2.5 Breaking Changes](#)

- **KPI Changes**
  - [KPI Changes from PCF v2.4 to v2.5](#)

- **Diego Network Communications**
  - [Diego Network Communications](#)

Migrate Apps to cflinuxfs3

In PAS v2.5, `cflinuxfs2` is no longer supported. Before upgrading to PAS v2.5, developers must change their stack to `cflinuxfs3` and restage their apps.

To prepare for your upgrade, do the following:

1. Install the Stack Auditor plugin for the cf CLI. See [Install Stack Auditor](#) in Using the Stack Auditor Plugin.

2. Audit stack usage to determine which apps need to be migrated from `cflinuxfs2` to `cflinuxfs3`. You can list apps and their stacks for each org you have access to by running the following command. To see all the apps in your deployment, ensure that you are logged in to the cf CLI as a user who can access all orgs.

   ```
cf audit-stack
   ```

   See the following example output:
3. Communicate to developers that they must migrate their existing apps to \texttt{cflinuxfs3} and begin pushing all new apps on \texttt{cflinuxfs3}. When changing stacks, developers may see errors related to new or changed libraries. If they do, they must update their app accordingly. Developers can migrate their apps using two methods:

- \texttt{cf push}: See Restaging Apps on a New Stack \texttt{\textsuperscript{[a]}} in Changing Stacks.
- \texttt{cf change-stack}: See Change Stacks \texttt{\textsuperscript{[a]}} in Using the Stack Auditor Plugin. This method does not require the source code of the app.

  \textbf{Note:} When changing the stack, the app experiences downtime. Pivotal recommends changing the stack using a zero-downtime strategy or at a time when you feel downtime is acceptable.

4. Confirm that there are no apps running on \texttt{cflinuxfs2} using the \texttt{cf audit-stack} command.

5. Delete all buildpacks in your deployment that are associated with \texttt{cflinuxfs2}.

6. Delete the \texttt{cflinuxfs2} stack. This action cannot be undone, with the following exception: If you delete \texttt{cflinuxfs2} and update to a later patch release of PAS v2.4, the stack returns and you must delete it again before upgrading to PAS v2.5.

   a. Run the following command:

   ```bash
   cf delete-stack cflinuxfs2
   ```

   b. When prompted, type \texttt{cflinuxfs2} and press enter.

   See the following example:

   ```bash
   $ cf delete-stack cflinuxfs2
   Are you sure you want to remove the cflinuxfs2 stack? If so, type the name of the stack [cflinuxfs2] >cflinuxfs2
   Deleting stack cflinuxfs2...
   Stack cflinuxfs2 has been deleted.
   
   If you have any apps still running on \texttt{cflinuxfs2}, the command returns the following error:
   ```
   Failed to delete stack cflinuxfs2 with error: Please delete the app associations for your stack.
   ```

---

**Update Tiles and Add-Ons**

The following section describes changes you must make to your product tiles and add-ons before upgrading PCF.

**Review Tile Compatibility**

Before you upgrade to PCF v2.5, check whether the service tiles that you currently have deployed on PCF v2.4 are compatible with PCF v2.5.

To check PCF versions supported by a service tile, either from Pivotal or a Pivotal partner:

- Navigate to the tile's download page on Pivotal Network \texttt{\textsuperscript{[b]}}.
- Select the tile version in the Releases dropdown.
- See the Depends On section under Release Details. For more information, refer to the tile's release notes.

If the currently-deployed version of a tile is not compatible with PCF v2.5, you must upgrade the tile to a compatible version before you upgrade PCF. You do not need to upgrade tiles that are compatible with both PCF v2.4 and v2.5.

Some partner service tiles may be incompatible with PCF v2.5. Pivotal works with partners to ensure their tiles are updated to work with the latest versions of PCF. For more information about which partner service release compatibility, review the Depends On section of the partners tile download page, the partners services release documentation in Pivotal Documentation \texttt{\textsuperscript{[b]}}, or contact the partner organization that produces the service tile.
The Product Compatibility Matrix provides an overview of which PCF versions support which versions of the most popular service tiles from Pivotal.

Environment Details

Pivotal provides the empty table below as a model to print out or adapt for recording and tracking the tile versions that you have deployed in all of your environments.

<table>
<thead>
<tr>
<th>Service</th>
<th>Sandbox</th>
<th>Non-Prod</th>
<th>Prod</th>
<th>Other…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pivotal Cloud Foundry</td>
<td>Ops Manager</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pivotal Cloud Foundry Services</td>
<td>MySQL v2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Redis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RabbitMQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Single Sign On</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spring Cloud Services</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concourse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pivotal Cloud Foundry Partner Services</td>
<td>New Relic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>…</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Upgrade Services Tiles

Upgrade all service tiles to versions that are compatible with PCF v2.5. Service tiles are add-on products you install alongside your runtime. For example, MySQL for PCF, PCF Healthwatch, and RabbitMQ are service tiles.

Do not upgrade runtime tiles, such as PAS, PASW, or Pivotal Container Service (PKS), at this time.

Review the Compatibility Matrix and tile documentation to check version compatibility.

Install BOSH CLI v5.4.0 or Later

Install BOSH CLI v5.4.0 to avoid IP conflict errors that cause BOSH healthcheck tasks to fail to acquire locks. For more information, see BOSH health check tasks fails to acquire lock in the Pivotal Knowledge Database.

Configure BOSH Director

With each release of a new PCF version, BOSH Director may require specific updates before upgrading to the new version. See the following for what action to take before upgrading to PCF v2.5:

Check Required Machine Specifications

Check the required machine specifications for Ops Manager v2.5. These specifications are specific to your IaaS. If these specifications do not match your existing Ops Manager, modify the values of your Ops Manager VM instance. For example, if the boot disk of your existing Ops Manager is 50 GB and the new Ops Manager requires 100 GB, then increase the size of your Ops Manager boot disk to 100 GB.

Ensure CA is Trusted by BOSH Director

Starting in PAS v2.4.3, Apps Manager verifies SSL certificates for endpoints to which it proxies. For environments using self-signed certificates or certificates that are signed by a certificate authority that is not trusted by the BOSH Director, this may cause Apps Manager to show no content.

If you are upgrading to PAS v2.5 from PAS v2.4.2 or earlier, you can prevent this issue by adding the certificate authority using the Trusted Certificates field in the Security pane of the BOSH Director tile.
Configure PAS

With each release of a new PCF version, PAS may require specific updates before upgrading to the new version. See the following sections for what action to take before upgrading to PCF v2.5:

Deselect Option to Override Deployment Name to CF

Ensure that the **Use “cf” as deployment name in emitted metrics instead of unique name** option in the Advanced Features pane of the PAS tile is deselected. This is a requirement to upgrade successfully.

Additionally, if you have PCF Healthwatch installed and you changed the value of **Use “cf” as deployment name in emitted metrics instead of unique name** in PAS, you must run the Push Monitoring Components errand for Healthwatch to detect the change.

For more information, see [Removed Option to Override Deployment Name to CF](#).

Configure Diego Cell Garbage collection

In the PAS tile Application Container pane, if **Docker Images Disk-Cleanup Scheduling on Cell VMs is set to Clean up disk-space once threshold is reached** and the value of **Threshold of Disk-Used (MB)** below has been changed from the default of **10240**, then operators need to pick a new threshold. If the scheduling is set to **Never clean up…** or **Routinely clean up…***, or the threshold value is set to the default, then the no action is necessary, and any threshold will migrate to a sensible value. For more information see [Options for Disk Cleanup](#).

Configure Gorouter with TLS

Before upgrading to PAS v2.5, you must secure the Gorouter with TLS or mutual TLS for PAS and Isolation Segment tiles.

The following sections describe how to enable routing with TLS or mutual TLS and scale the Diego cell VM CPU and RAM. If you do not have enough RAM on your Diego cell VM after enabling TLS routing, you are unable to stage tasks and app instances. App instances may also stop running.

**Note:** Gorouter with TLS or mutual TLS is not supported in the PASW tile.

**Step 1: Enable TLS or Mutual TLS Routing**

To enable TLS or mutual TLS routing, do the following:

1. From the **Ops Manager Installation Dashboard**, go to the PAS tile.
2. Go to the **Application Containers** pane.
3. Under **Router application identity verification**, select either of the following options:
   - **Router uses TLS to verify application identity**
   - **Router and applications use mutual TLS to verify each other’s identity**
4. Click **Save**.

**Step 2: Determine Number of App Instances**

Before you scale your Diego cell VM to handle TLS routing, you must determine the number of app instances running on your deployment.

See the following methods for how you can count your app instances. Choose the method that corresponds to your use case.

1. **Deployments without Isolation Segments**
   1. Access your platform metrics with your configured monitoring tool or with cf CLI Firehose nozzle plugin. For more information about the CLI Firehose nozzle plugin, see [Installing the Loggregator Firehose Plugin for cf CLI](#). For more information about configuring a monitoring system, see [Selecting and Configuring a Monitoring System](#).
   2. Find the **LRPDesired** metric of the **lbs** job on the Diego Database VM. See the following example:
3. Record the \texttt{value} of the \texttt{LRPsDesired} for each instance of the \texttt{bbs} job. You need the \texttt{value} for the procedure in the next section.

### Deployments with Isolation Segments

1. Access your platform metrics with your configured monitoring tool or with cf CLI Firehose nozzle plugin. For more information about the CLI Firehose nozzle plugin, see \textit{Installing the Loggregator Firehose Plugin for cf CLI}. For more information about configuring a monitoring system, see \textit{Selecting and Configuring a Monitoring System}.

2. Find all \texttt{ContainerCount} metrics of the \texttt{rep} job of each Diego cell VM. See the following example:

```plaintext
origin: "rep" eventType: ValueMetric
timestamp: 15415419000289448
deployment: "cf"
job: "diego_cell"
index: "8007afda-3bff-4856-857f-a47a43cbf994"
ip: "10.0.4.18"
tags:<
key: "instance_id"
value: "8007afda-3bff-4856-857f-a47a43cbf994"
>
valueMetric:<
name: "ContainerCount"
value: 200
unit: "Metric"
>
```

3. For each \texttt{ContainerCount} metric, record the \texttt{value} of the \texttt{ContainerCount} and the \texttt{ip} of the \texttt{job}.

#### Step 3: Scale Diego Cell VM

To support TLS or mutual TLS routing, you must have enough CPU and RAM for your Diego cell VM. TLS routing requires an additional 32 MB of RAM on your Diego cell per app instance.

To calculate and configure the amount of RAM you need for your Diego cell, choose one of the following methods for your use case:

### Deployments without Isolation Segments

For your PAS tile, do the following:

1. Go to the \textit{Resource Config} pane.
2. In the \textit{Diego Cell} row, see your current \textbf{VM Type} with the amount of RAM you currently have.
3. Multiply the \texttt{value} you recorded in the previous section by 32. Add your solution to the amount of RAM you currently have.
4. Select your new \textbf{VM Type} based on the amount of RAM you need.
5. Click \textit{Save}.

### Deployments with Isolation Segments

For each PAS and Isolation Segment tile in your foundation, do the following:

1. Go to the \textit{Status} tab and see the IP of your Diego cell. To determine the \texttt{value} that corresponds to this tile, match the IP to the \texttt{ip} metric you recorded in the previous section.
2. Go to the \textit{Resource Config} pane of the tile.
3. In the \textit{Diego Cell} row, see your current \textbf{VM Type} with the amount of RAM you currently have.
4. Multiply the \texttt{value} for this tile by 32. Add your solution to the amount of RAM you currently have.
5. Select your new \textbf{VM Type} based on the amount of RAM you need.
6. Click \textit{Save}.

### (Optional) Migrate from Metrics Forwarder to Metric Registrar

The Metric Registrar allows app developers to export custom app metrics to Loggregator. Metrics Forwarder for PCF is a service that also allows apps to emit custom metrics.

Metrics Forwarder for PCF is not supported in PAS v2.5. If you use Metrics Forwarder for PCF to emit custom app metrics to Loggregator, Pivotal recommends enabling and configuring the Metric Registrar before upgrading to PAS v2.5. The PAS tile does not deploy the Metric Registrar by default.

To enable the Metric Registrar, do the following:

1. In the PAS tile, click \textit{Metric Registrar}.
2. Select the \textit{Enable Metric Registrar} checkbox.

For more information about enabling the Metric Registrar in PAS, see \textit{(Optional) Configure Metric Registrar} in Configuring PAS.

For more information about the Metric Registrar, see \textit{Registering Custom App Metrics}.

© Copyright Pivotal Software Inc, 2013-2019
(Optional) Disable Unused Errands

To save upgrade time, you can disable unused PAS post-deploy errands. See the Post-Deploy Errands section of the Errands topic for details. Only disable these errands if your environment does not need them.

In some cases, if you have previously disabled lifecycle errands for any installed product to reduce deployment time, you may want to re-enable these errands before upgrading. For more information, see the Adding and Deleting Products topic.

Check OS Compatibility of BOSH-Managed Add-Ons and Tiles

Before upgrading to PCF v2.5, operators who have deployed any PCF add-ons such as IPSec for PCF, ClamAV for PCF, or File Integrity Monitoring for PCF and who have deployed or are planning to deploy PASW must modify the add-on manifest to specify a compatible OS stemcell. For more information, see Pivotal Application Service for Windows.

For example, File Integrity Monitor for PCF (FIM) is not supported on Windows. Therefore, the manifest must use an include directive to specify the target OS stemcell of ubuntu-trusty and ubuntu-xenial.

Note: To upgrade to a Xenial stemcell, see the documentation for each add-on and follow the instructions.

To update an add-on manifest, do the following:

1. Locate your existing add-on manifest file. For example, for FIM, locate the fim.yml you uploaded to the Ops Manager VM.

2. Modify the manifest to include following include directive to your manifest:

   ```yaml
   include:
     stemcell:
       - os: ubuntu-xenial
   ```

3. Upload the modified manifest file to your PCF deployment. For example instructions, see Create the FIM Manifest.

If you use any other BOSH-managed add-ons in your deployment, you should verify OS compatibility for those component as well. For more information about configuring BOSH add-on manifests, see the BOSH documentation.

Check Backup and Restore External Blobstore Add-On

If you have enabled external blobstore backups for an Azure Blobstore using the Blobstore Add-On, you must update your runtime configuration to remove the sdk-preview add-on before upgrading to PCF v2.5. If you do not remove this job, upgrading PAS fails with the error:

```
Preparing deployment: Preparing deployment (00:00:01)
    Error: Colocated job 'azure-blobstore-backup-restorer' is already added to the instance group 'backup-restore'.
```

After removing this job from your runtime configuration, ensure that the Enable backup and restore checkbox is enabled in the PAS tile > File Storage pane.

Check Certificate Authority Expiration Dates

Depending on the requirements of your deployment, you may need to rotate your Certificate Authority (CA) certificates. The non-configurable certificates in your deployment expire every two years. You must regenerate and rotate them so that critical components do not face a complete outage.

Note: PCF uses SHA-2 certificates and hashes by default. You can convert existing SHA-1 hashes into SHA-2 hashes by rotating your Ops Manager certificates using the procedure described in the Regenerating and Rotating Non-Configurable TLS/SSL Certificates section of Managing TLS Certificates.

To retrieve information about all the RSA and CA certificates for the BOSH Director and other products in your deployment, you can use the GET api/v0/deployed/certificates?expires_within=TIME request of the Ops Manager API.

In this request, the expires_within parameter is optional. Valid values for the parameter are: d for days, w for weeks, m for months, and y for years.

For example, to search for certificates expiring within one month, replace TIME with 1m.
For information about regenerating and rotating CA certificates, see Managing TLS Certificates.

Check the Capacity of Your Deployment

The following sections describe steps for ensuring your deployment has adequate capacity to perform the upgrade.

Confirm Adequate Disk Space

Confirm that you have adequate disk space for your upgrades. You need at least 20 GB of free disk space to upgrade PCF Ops Manager and Pivotal Application Service. If you plan to upgrade other products, the amount of disk space required depends on how many tiles you plan to deploy to your upgraded PCF deployment.

To check current persistent disk usage, select the BOSH Director tile from the Installation Dashboard. Select Status and review the value of the PERS.DISK column. If persistent disk usage is higher than 50%, select Settings > Resource Config, and increase your persistent disk space to handle the size of the resources. If you do not know how much disk space to allocate, set the value to at least 100 GB.

Check Diego Cell RAM and Disk

Check that Diego cells have sufficient available RAM and disk capacity to support app containers.

The KPIs that monitor these resources are:

- rep.CapacityRemainingMemory
- rep.CapacityRemainingDisk

Adjust Diego Cell Limits

If needed, adjust the maximum number of Diego cells that the platform can upgrade simultaneously, to avoid overloading the other cells. See Limit PCF Component Instances During Restart.

For PCF v1.10 and later, the maximum number of cells that can update at once, max_in_flight is 4%. This setting is configured in the BOSH manifest's Diego cell definition. For more information, see the Prevent Overload section.

Review the Diego Cell Metrics section of the KPI topic for more information about these KPIs.

Review File Storage IOPS and Other Upgrade Limiting Factors

During the PCF upgrade process, a large quantity of data is moved around on disk.

To ensure a successful upgrade of PCF, verify that your underlying PAS file storage is performant enough to handle the upgrade. For more information about the configurations to evaluate, see Upgrade Considerations for Selecting Pivotal Cloud Foundry Storage.

In addition to file storage IOPS, consider additional existing deployment factors that can impact overall upgrade duration and performance:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network latency</td>
<td>Network latency can contribute to how long it takes to move app instance data to new containers.</td>
</tr>
<tr>
<td>Number of ASGs</td>
<td>A large number of Application Security Groups in your deployment can contribute to an increase in app instance container startup time.</td>
</tr>
<tr>
<td>Number of app instances and application growth</td>
<td>A large increase in the number of app instances and average droplet size since the initial deployment can increase the upgrade impact on your system.</td>
</tr>
</tbody>
</table>

To review example upgrade-related performance measurements of an existing production Cloud Foundry deployment, see the Pivotal Web Services.
Performance During Upgrade topic.

Run BOSH Clean-Up

Run `bosh -e ALIAS clean-up --all` to clean up old stemcells, releases, orphaned disks, and other resources before upgrade. This cleanup helps prevent the product and stemcell upload process from exceeding the BOSH Director's available persistent disk space.

Check the Health of Your Deployment

The following sections describe steps for ensuring your deployment is healthy before you perform the upgrade.

Collect Foundation Health Status

For collecting foundation health status, Pivotal recommends PCF Healthwatch, which monitors and alerts on the current health, performance, and capacity of PCF. For more information, see the [PCF Healthwatch documentation](#).

If you are not using PCF Healthwatch, you can do some or all of the following to collect foundation health status:

- If your PCF deployment has external metrics monitoring set up, verify that VM CPU, RAM, and disk use levels are within reasonable levels.
- Run BOSH CLI commands to check system status:
  - `bosh -e ALIAS -d DEPLOYMENT_NAME instances --ps`.
  - `bosh -e ALIAS vms --vitals`. This reveals VMs with high CPU, high memory, high disk utilization, and with `state != running`.
  - `bosh -e ALIAS -d DEPLOYMENT_NAME cck --report`
- Check Ops Manager GUI each PAS/Tiles the status page for CPU/RAM/DISK utilization
- Validate Ops Manager persistent disk usage is below 50%. If not, follow the procedure in [Confirm Adequate Disk Space](#).

(Optional) Check the logs for errors before proceeding with the upgrade. For more information, see [Viewing Logs in the Command Line Interface](#).

Push and Scale a Test App

Check that a test app can be pushed and scaled horizontally, manually or through automated testing. This check ensures that the platform supports apps as expected before the upgrade.

Validate MySQL Cluster Health

If you are running PAS MySQL as a cluster, run the `mysql-diag` tool to validate health of the cluster.

See the BOSH CLI v2 instructions in the [Running mysql-diag](#) topic.

Review Pending and Recent Changes

1. Confirm there are no outstanding changes in Ops Manager or any other tile. All tiles should be green. Click Review Pending Changes, then Apply Changes if necessary.

2. After applying changes, click Recent Install Logs to confirm that the changes completed cleanly:

```
Cleanup complete
("type": "step_finished", "id": "clean_up_bosh.cleaning_up")
Exited with 0.
```

© Copyright Pivotal Software Inc, 2013-2019
Export Your Installation

To export your installation, do the following:

1. In your Ops Manager v2.4 Installation Dashboard, click the account dropdown and select **Settings**.

2. On the **Settings** screen, select **Export Installation Settings** from the left menu, then click **Export Installation Settings**.

This exports the current PCF installation with all of its assets.
When you export an installation, the export contains the base VM images, necessary packages, and configuration settings, but does not include releases between upgrades if Ops Manager has already uploaded them to BOSH. When backing up PCF, you must take this into account by backing up the BOSH blobstore that contains the uploaded releases. BOSH Backup and Restore (BBR) backs up the BOSH blobstore. For more information, see Backing Up Pivotal Cloud Foundry with BBR.

- The export time depends on the size of the exported file.
- Some browsers do not provide feedback on the status of the export process and might appear to hang.

**Note:** Some operating systems automatically unzip the exported installation. If this occurs, create a ZIP file of the unzipped export. Do not start compressing at the “installation” folder level. Instead, start compressing at the level containing the `config.yml` file:

![Compress 7 items](image)

**warning:** If you fail to perform the remedial steps for this issue, this upgrade process may corrupt your existing usage data.

### Next Steps

Now that you have completed the Upgrade Preparation Checklist for PCF v2.5, continue to Upgrading Pivotal Cloud Foundry.

### Complete Survey

Please take some time to help us improve this document by completing the Upgrade Checklist Survey.
Configuring PAS for Upgrades

This topic describes several configuration options for Pivotal Application Service (PAS) that can help ensure successful upgrades. In addition to following the Upgrade Preparation Checklist, review the sections in this document to better understand how to prepare for PAS Upgrades.

Limit PCF Component Instance Restarts

The `max_in_flight` variable limits how many instances of a component can restart simultaneously during updates or upgrades. Increasing the value of `max_in_flight` can make updates run faster, but setting it too high risks overloading VMs and causing failure. See Best Practices for guidance on setting `max_in_flight` values.

Values for `max_in_flight` can be any integer between 1 and 100, or a percentage of the total number of instances. For example, a `max_in_flight` value of 20% in a deployment with 10 Diego cell instances would make no more than two cell instances restart at once.

Set max_in_flight

The `max_in_flight` variable is a system-wide value with optional component-specific overrides. You can override the default value for individual jobs using an API endpoint.

Use the max_in_flight API Endpoint

Use the `max_in_flight` API endpoint to configure the maximum value for component instances that can start at a given time. This endpoint overrides product defaults. You can specify values as a percentage or an integer.

Use the string “default” as the `max_in_flight` value to force the component to use the deployment’s default value.

```
curl "https://EXAMPLE.com/api/v0/staged/products/PRODUCT-TYPE1-GUID/max_in_flight"
-X PUT
-H "Authorization: Bearer UAA_ACCESS_TOKEN"
-H "Content-Type: application/json"
-d '{
  "max_in_flight": {
    "JOB_1_GUID": 1,
    "JOB_2_GUID": "20%",
    "JOB_3_GUID": "default"
  }
}'
```

Note: The example below lists three JOB_GUIDs. These three GUIDs are examples of the three different types of values you can use to configure `max_in_flight`. The endpoint only requires one GUID.

Specific Guidance for Diego Cells

To upgrade Cloud Foundry, BOSH must drain all Diego cell VMs that host app instances. BOSH manages this process by upgrading a batch of cells at a time.

The number of cells that undergo upgrade simultaneously (either in a state of shutting down or coming back online) is controlled by the `max_in_flight` value of the Diego cell job. For example, if `max_in_flight` is set to 10% and your deployment has 20 Diego cell job instances, then the maximum number of cells that BOSH can upgrade at a single time is 2.

When BOSH triggers an upgrade, each Diego cell undergoing upgrade enters “evacuation” mode. Evacuation mode means that the cell stops accepting new work and signals the rest of the Diego system to schedule replacements for its app instances. This scheduling is managed by the Diego auctioneer process.

The evacuating cells continue to interact with the Diego system as replacements come online. The cell undergoing upgrade waits until either its app instance replacements run successfully before shutting down the original local instances, or for the evacuation process to time out. This “evacuation timeout” defaults to 10 minutes.

If cell evacuation exceeds this timeout, then the cell stops its app instances and shuts down. The Diego system continues to re-emit start requests for the
Prevent Overload

A potential issue arises if too many app instance replacements are slow to start or do not start successfully at all.

If too many app instances are starting concurrently, then the load of these starts on the rest of the system can cause other applications that are already running to crash and be rescheduled. These events can result in a cascading failure.

To prevent this issue, PCF provides two throttle configurations: the maximum number of in-flight diego cell instances and the maximum number of starting containers.

The values of the above throttle configurations depend on the version of PCF that you have deployed and whether you have overridden the default values.

Refer to the following table for existing defaults and, if necessary, determine the override values in your deployment.

<table>
<thead>
<tr>
<th>PCF Version</th>
<th>Starting Container Count Maximum</th>
<th>Starting Container Count Overridable?</th>
<th>Maximum In Flight Diego Cell Instances</th>
<th>Maximum In Flight Diego Cell Instances Overridable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCF 1.7.43 and earlier</td>
<td>No limit set</td>
<td>No</td>
<td>1 instance</td>
<td>No</td>
</tr>
<tr>
<td>PCF 1.7.44 to 1.7.49</td>
<td>200</td>
<td>No</td>
<td>1 instance</td>
<td>No</td>
</tr>
<tr>
<td>PCF 1.7.50 +</td>
<td>200</td>
<td>No</td>
<td>1 instance</td>
<td>No</td>
</tr>
<tr>
<td>PCF 1.8.0 to 1.8.29</td>
<td>200</td>
<td>No</td>
<td>10% of total instances</td>
<td>No</td>
</tr>
<tr>
<td>PCF 1.8.30 +</td>
<td>No limit set</td>
<td>Yes</td>
<td>10% of total instances</td>
<td>No</td>
</tr>
<tr>
<td>PCF 1.9.0 to 1.9.7</td>
<td>No limit set</td>
<td>No</td>
<td>4% of total instances</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF 1.9.8 +</td>
<td>200</td>
<td>Yes</td>
<td>4% of total instances</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF 1.10.0 and later</td>
<td>200</td>
<td>Yes</td>
<td>4% of total instances</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF 1.12.0 and later</td>
<td>200</td>
<td>Yes</td>
<td>4% of total instances</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Best Practices

Set the `max_in_flight` variable high enough that the remaining component instances are not overloaded by typical use. If component instances are overloaded during updates, upgrades, or typical use, users may experience downtime.

Some more precise guidelines include:

- For jobs with high resource usage, set `max_in_flight` low. For example, for Diego cells, `max_in_flight` allows non-migrating cells to pick up the work of cells stopping and restarting during migration. If resource usage is already close to 100%, scale up your jobs before making any updates.

- For quorum-based components (these are components with odd-numbered settings in the manifest), such as etcd, consul, and diego BBS, set `max_in_flight` to 1. This preserves quorum and prevents a split-brain scenario from occurring as jobs restart.

- For other components, set `max_in_flight` to the number of instances that you can afford to have down at any one time. The best values for your deployment vary based on your capacity planning. In a highly redundant deployment, you can make the number high so that updates run faster. If your components are at high utilization, however, you should keep the number low to prevent downtime.

- Never set `max_in_flight` to a value greater than or equal to the number of instances you have running for a component.

Set a Maximum Number of Starting Containers

This section describes how to use the auctioneer job to configure the maximum number of app instances starting at a given time. This prevents Diego from scheduling too much new work for your platform to handle concurrently. A lower default can prevent server overload during cold start, which may be important if your infrastructure is not sized for a large number of concurrent cold starts.

The auctioneer only schedules a fixed number of app instances to start concurrently. This limit applies to both single and multiple Diego Cells. For
example, if you set the limit to five starting instances, it does not matter if you have one Diego Cell with ten instances or five Diego Cells with two instances each. The auctioneer will not allow more than five instances to start at the same time.

If you are using a cloud-based IaaS, rather than a smaller on-premise solution, Pivotal recommends setting a larger default. By default, the maximum number of started instances is 200.

You can configure the maximum number of started instances in the Settings tab of the Pivotal Application Service (PAS) tile.

1. Log in to Operations Manager.
2. Click the PAS tile.
3. Click Application instances in the sidebar.
4. In the Max Inflight Container Starts field, type the maximum number of started instances.

5. Click Save.

Configure File Storage

This section describes critical factors to consider when evaluating the type of file storage to use in your Pivotal Cloud Foundry (PCF) deployment. The Pivotal Application Service (PAS) blobstore relies on the file storage system to read and write resources, app packages, and droplets.

During an upgrade of PCF, file storage with insufficient IOPS numbers can negatively impact the performance and stability of your PCF deployment.

If disk processing time takes longer than the evacuation timeout for Diego cells, then Diego cells and app instances may take too long to start up, resulting in a cascading failure.

However, the minimum required IOPS depends upon a number of deployment-specific factors and configuration choices. Use this section as a guide when deciding on the file storage configuration for your deployment.

To see an example of system performance and IOPS load during an upgrade, refer to Upgrade Load Example: Pivotal Web Services.

Select Internal or External File Storage

When you deploy PCF, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the PAS
Selecting internal storage causes PCF to deploy a dedicated virtual machine (VM) that uses either NFS or WebDAV for file storage. Selecting external storage allows you to configure file storage provided in network-accessible location or by an IaaS, such as Amazon S3, Google Cloud Storage, or Azure Storage.

Whenever possible, Pivotal recommends using external file storage.

Calculate Potential Disk Load Requirements

As a best-effort calculation, estimate the total number of bits needed to move during a system upgrade to determine how IOPS-performant your file storage needs to be.

- **Number of Diego Cells**
  
  - As a first calculation, determine the number of Diego cells that your deployment currently uses. To view the number of Diego cell instances currently running in your deployment, see the Resource Config section of your PAS tile. If you expect to scale up the number of instances, use the anticipated scaled number.

    Note: If your deployment uses more than 20 Diego cells, you should avoid using internal file storage. Instead, you should always select external or IaaS-provided file storage.

- **Maximum In-Flight Load and Container Starts for Diego Cells**
  
  - Operators can limit the number of containers and Diego cell instances that Diego starts concurrently. If operators impose no limits, your file storage may experience exceptionally heavy load during an upgrade.
Upgrade Load Example: Pivotal Web Services

This topic provides sample performance measurements of a Cloud Foundry installation undergoing the workload associated with an upgrade.

To obtain these measurements, Pivotal repaved its production Pivotal Web Services (PWS) deployment. The repave process simulates system load that would be incurred when performing a rolling upgrade of Diego cells.

Use the measurements and configuration values published in this document as guidance when ensuring you have adequate file storage hardware prior to a platform upgrade.

For more information about the impact of upgrade on file storage performance, see Configure File Storage.

Platform Configuration

The following table details the starting parameters and configuration of PWS.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
<th>How to Locate</th>
</tr>
</thead>
<tbody>
<tr>
<td>IaaS</td>
<td>Amazon Web Services</td>
<td>Refer to your BOSH Director configuration or BOSH deployment manifest.</td>
</tr>
<tr>
<td>File Storage</td>
<td>AWS EBS (External with some elastic capacity)</td>
<td>Refer to your Pivotal Application Service (PAS) configuration or BOSH deployment manifest.</td>
</tr>
<tr>
<td>Version of CF</td>
<td>v252</td>
<td>Refer to your BOSH Director and PAS configuration or BOSH deployment manifest.</td>
</tr>
<tr>
<td>Number of Diego Cells</td>
<td>218</td>
<td>To view the number of Diego cell instances currently running in your deployment, see the Resource Config section of your PAS tile or consult your Diego deployment manifest.</td>
</tr>
<tr>
<td>Maximum Number of Started Containers</td>
<td>250</td>
<td>See Configuring PAS for Upgrades.</td>
</tr>
<tr>
<td>max_in_flight</td>
<td>6</td>
<td>To retrieve the existing max_in_flight value for the Diego Cell job in BOSH Director, use the Ops Manager API. See the Ops Manager API documentation. If you are running open source CF, consult your BOSH deployment manifest.</td>
</tr>
<tr>
<td>Number of Availability Zones (AZ)</td>
<td>2</td>
<td>Consult your PAS or BOSH deployment AZ configuration.</td>
</tr>
<tr>
<td>Number of App Instances</td>
<td>16231</td>
<td><a href="#">datadog.nozzle.bbs.LRPsRunning</a></td>
</tr>
<tr>
<td>Number of Application Security Groups (ASGs)</td>
<td>43</td>
<td>As admin user, run the <code>cf security-groups</code> command. For more information, see Application Security Groups.</td>
</tr>
</tbody>
</table>

System Performance Measurements During Cell Repave

This table presents performance measurements taken during the Diego cell repave.

![Note: These measurements indicate the peak cumulative values of the entire system (250 Diego cells, ~15,000 application instances, and 2 AZs.)](#)

Use these measurements as a baseline for expected system load during Diego cell upgrade.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Value</th>
<th>Metric Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell CPU Consumption</td>
<td>36%</td>
<td><code>bosh.healthmonitor.system.cpu.use</code></td>
</tr>
<tr>
<td>Cell Memory Consumption</td>
<td>~50%</td>
<td><code>bosh.healthmonitor.system.mem.percent</code></td>
</tr>
<tr>
<td>Cell I/O Consumption (Read) During Normal Operations</td>
<td>43 Read I/O Operations per second</td>
<td><code>aws.ebs.volume_read_ops</code></td>
</tr>
<tr>
<td>Cell I/O Consumption (Read) During Upgrade</td>
<td>1,943 Read I/O Operations per second</td>
<td>aws.ebs.volume_read_ops</td>
</tr>
<tr>
<td>Cell IO Consumption (Write) During Normal Operations</td>
<td>2,166 Write I/O Operations per second</td>
<td>aws.ebs.volume_write_ops</td>
</tr>
<tr>
<td>Cell IO Consumption (Write) During Upgrade</td>
<td>21,000 Write I/O Operations per second</td>
<td>aws.ebs.volume_write_ops</td>
</tr>
<tr>
<td>Cell Network Consumption (Network Out) During Normal Operations</td>
<td>~1.25 GB per minute</td>
<td>aws.ec2.network_out</td>
</tr>
<tr>
<td>Cell Network Consumption (Network Out) During Upgrade</td>
<td>~1.25 GB per minute (no significant change)</td>
<td>aws.ec2.network_out</td>
</tr>
<tr>
<td>Cell Network Consumption (Network In) During Normal Operations</td>
<td>2.11 GB per minute</td>
<td>aws.ec2.network_in</td>
</tr>
<tr>
<td>Cell Network Consumption (Network In) During Upgrade</td>
<td>16.75 GB per minute</td>
<td>aws.ec2.network_in</td>
</tr>
</tbody>
</table>

Sample Performance Graphs

These DataDog graphs represent a timeline visualization of read and write operations during the repave event.

Read I/O Operations

The read I/O operations sample was taken over 115 VMs and represent the number of read operations over 300 seconds for a single Diego cell.

Write I/O Operations

The write I/O operations sample was taken over 115 VMs and represent the number of write I/O operations over 300 seconds for a single Diego cell.

Summary

During the repave process, 250 Diego cells were updated. The repave process took 6 hours overall or about 3 hours for each Availability Zone.
Upgrading PAS and Other Pivotal Cloud Foundry Products

This topic describes how to upgrade to a point release of Pivotal Application Service (PAS) and other product tiles without upgrading Ops Manager. For example, use this topic to upgrade from PAS v1.9.0 to v1.9.1. You might need to perform this upgrade if a security update for PAS is released, or if new features are introduced in a point release of a product tile.

For PAS component and version information, see the PAS release notes.

Note: If you cannot download products from Pivotal Network due to restricted network connectivity, see Installing PCF in Airgapped Environments.

Before You Upgrade to Point Releases

- You must have completed the Upgrading Pivotal Cloud Foundry procedure.
- Refer to the Product Compatibility Matrix before upgrading PAS.
- Important: Read the Known Issues sections of the products you plan on installing before starting. See Pivotal Cloud Foundry Release Notes for all available product release notes.

Upgrading PAS

Note: If you are using the Pivotal Network API, the latest product versions will automatically appear in your Installation Dashboard.

To upgrade PAS without upgrading Ops Manager, do the following:

1. Download the product file from Pivotal Network.
2. Import the product file to your Ops Manager Installation Dashboard.
3. Click the plus icon next to the uploaded product description to add this product to your staging area.
4. Click the newly added tile to review any configurable options.
5. Click Review Pending Changes, then Apply Changes to install the service.

Upgrading PCF Products

Note: If you are using the Pivotal Network API, the latest product versions will automatically appear in your Installation Dashboard.

This section describes how to upgrade individual products like Single Sign-On for PCF, MySQL for PCF, RabbitMQ for PCF, and Metrics for PCF in your Pivotal Cloud Foundry (PCF) deployment. Ensure you review the individual product upgrade procedure for each tile.

1. Browse to Pivotal Network and sign in.
2. Download the latest PCF release for the product or products you want to upgrade. Every product is tied to exactly one stemcell. Download the stemcell that matches your product and version.
3. Confirm that you have adequate disk space for your upgrades. You need at least 20 GB of free disk space to upgrade PCF Ops Manager and Pivotal Application Service. If you plan to upgrade other products, the amount of disk space required depends on how many tiles you plan to deploy to your upgraded PCF deployment.

To check current persistent disk usage, select the BOSH Director tile from the Installation Dashboard. Select Status and review the value of the PERS_DISK column. If persistent disk usage is higher than 50%, select Settings > Resource Config, and increase your persistent disk space to handle the size of the resources. If you do not know how much disk space to allocate, set the value to at least 100 GB.

4. Browse to the Pivotal Cloud Foundry Ops Manager web interface and click Import a Product.
5. Select the .pivotal file that you downloaded from Pivotal Network or received from your software distributor, then click Open. If the product is successfully added, it appears in your product list. If the product that you selected is not the latest version, the most up-to-date version will appear on your product list.

6. Click the plus icon next to the product description to add the product tile to the Installation Dashboard.

7. Repeat the import, upload, and upgrade steps for each product you downloaded.

8. If you are upgrading a product that uses a self-signed certificate from v1.1 to v1.2, you must configure the product to trust the self-signed certificate.
   To configure a product to trust the self-signed certificate, do the following:
   a. Click the product tile.
   b. In the left-hand column, select the setting page containing the SSL certificate configuration. For example, for PAS, select the HAProxy page.
   c. Check the Trust Self-Signed Certificates box.
   d. Click Save.

9. Click Review Pending Changes, then Apply Changes.

### Upgrading Replicated Tiles

To upgrade a replicated tile, you must do the following:

- Generate a replica of the newer version of the tile using the replicator
- Give the new replica the same name as the existing replica.

See the following example workflow.

#### Example Workflow

This example assumes the following:

- You used the replicator to generate a replica of v1 of the Isolation Segments tile.
- You used the `--name` `seg-one`.
- You installed the tile in Ops Manager.

Here is the sample replicator command you used for the initial installation:

```
./replicator-darwin
--name seg-one
--path /download/p-isolation-segment-v1.pivotal
--output /output/p-isolation-segment-v1-seg-one.pivotal
```

To upgrade to v2, follow these steps:

1. Download and unzip the new Isolation Segment Tile Replicator from Pivotal Network. You must download the version of Tile Replicator that corresponds with the version of the Isolation Segment Tile you want to replicate.

2. Run the replicator command below to create the replica:
Where:

- **NAME-OF-EXISTING-REPLICA** must be the same as the name used for the existing replica. This is **seg-one** in this example.
- **PATH-TO-NEW TILE** is the path to the new Isolation Segment Tile.

3. After you have the replica tile **p-isolation-segment-v2-seg-one.pivotal**, upload it to Ops Manager. This upgrades the v1 replica tile in place.
cf push Availability During Pivotal Application Service Upgrades

Page last updated:

This topic describes what you can expect regarding the availability of `cf push` during upgrades of Pivotal Application Service (PAS).

**Overview**

Availability of `cf push` during PAS upgrades varies from release to release. There are various considerations, such as Cloud Controller database (CCDB) migrations or the number of VMs in use, that can impact `cf push` availability. However, `cf push` is mostly available for the duration of an upgrade.

**Impact on Single VMs vs. High-Availability Infrastructure**

Having a single VM in use, such as a WebDAV, a HAProxy, a Gorouter, a UAA, or a Cloud Controller, impacts whether `cf push` is unavailable during an upgrade.

If you have scaled out your application to achieve high availability and are not using WebDAV, `cf push` should be available for the entire duration of the upgrade. However, upgrades to certain versions of PAS sometimes require a CCDB schema or data migration, which may cause `cf push` to be unavailable while Cloud Controllers are rolling during the upgrade.

**Availability During a BBR Backup**

The Cloud Controller API is taken down during the `pre-backup-lock` stage of a BBR backup and put back up again during the `post-backup-unlock` stage. As a result, `cf push` becomes unavailable during that time. However, the backup that takes place between those stages is very short, no longer than a few minutes. The bulk of the BBR backup operation happens after the `post-backup-unlock` stage, so the Cloud Controller API and `cf push` are available for most of the duration of a BBR backup.

The Uptimer tool can help you measure `cf push` availability during an upgrade.
Platform Architecture and Planning

Introduction

A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as AWS, Azure, GCP, or vSphere, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

These documents detail PCF reference architectures for different IaaSes to help you determine the best configuration for your PCF deployment.

Products Covered by the Reference Architectures

Pivotal has validated the following PCF products on its own deployments based on these reference architectures:

- Pivotal Application Service (PAS)
- Pivotal Cloud Foundry Ops Manager
- Pivotal Container Services (PKS)

Available Platform Architectures and Planning Topics

- Pivotal Cloud Foundry on AWS
- Pivotal Cloud Foundry on Azure
- Pivotal Cloud Foundry on GCP
- Pivotal Cloud Foundry on OpenStack
- Pivotal Cloud Foundry on vSphere
- Control Plane Reference Architectures
- Implementing a Multi-Foundation PKS Deployment
- Using Global DNS Load Balancers for Multi-Foundation
This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on Amazon Web Services (AWS). This architecture is valid for most production-grade PCF deployments using three availability zones (AZs).

See [PCF on AWS Requirements](#) for general requirements for running PCF and specific requirements for running PCF on AWS.

### PCF Reference Architectures

A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as AWS, that meets the following requirements:

- **Secure**
- **Publicly-accessible**
- **Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services**
- **Can host at least 100 app instances, or far more**

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

### Base AWS Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on AWS using three AZs.

![Base AWS Reference Architecture](#)

**View a larger version of this diagram [here](#).**

**Note:** Each AWS subnet must reside entirely within one AZ. As a result, a multi-AZ deployment topology requires a subnet for each AZ.

### Base Reference Architecture Components

© Copyright Pivotal Software Inc, 2013-2019
The following table lists the components that are part of a base reference architecture deployment on AWS with three AZs.

<table>
<thead>
<tr>
<th>Component</th>
<th>Reference Architecture Notes</th>
</tr>
</thead>
</table>
| Domains & DNS | CF Domain Zones and routes in use by the reference architecture include:  
- domains for *.apps and *.sys (required)  
- a route for Ops Manager (required)  
- a route for ssh access to app containers (optional)  
Using Route 53 to manage domains is optional. |
| Ops Manager | Deployed on one of the three public subnets and accessible by FQDN or through an optional jumpbox. |
| BOSH Director | Deployed on the infrastructure subnet. |
| Elastic Load Balancers - HTTP, HTTPS, and SSL | Required. Load balancer that handles incoming HTTP, HTTPS, and SSL traffic and forwards them to the Gorouters. Deployed on all three public subnets. |
| Elastic Load Balancers - SSH | Optional. Load balancer that provides SSH access to app containers. Deployed on all three public subnets, one per AZ. |
| Gorouters | Accessed through the HTTP, HTTPS, and SSL Elastic Load Balancers. Deployed on all three Pivotal Application Service (PAS) subnets, one per AZ. |
| Diego Brains | Required. However, the SSH container access functionality is optional and enabled through the SSH Elastic Load Balancers. Deployed on all three PAS subnets, one per AZ. |
| TCP Routers | Optional feature for TCP routing. Deployed on all three PAS subnets, one per AZ. |
| CF Database | Reference architecture uses AWS RDS. Deployed on all three RDS subnets, one per AZ. |
| Storage Buckets | Reference architecture uses 4 S3 buckets: buildpacks, droplets, packages, and resources. |
| Service Tiles | Deployed on all three service subnets, one per AZ. |
| Service User & Roles | One IAM role and one IAM user are recommended: the IAM role for Terraform, and the IAM user for Ops Manager and BOSH. Consult the following list:  
- Admin Role: Terraform uses this IAM role to provision required AWS resources as well as an IAM user.  
- IAM User: This IAM user with IAM security credentials (access key ID and secret access key) is automatically provisioned with restrict access only to resources needed by PCF. |
| EC2 Instance Quota | The default EC2 instance quota on a new AWS subscription only has around 20 EC2 instances, which is not enough to host a multi-AZ deployment. The recommended quota for EC2 instances is 100. AWS requires the instances quota tickets to include Primary Instance Types, which should be t2.micro. |

Network Objects

The following table lists the network objects in this reference architecture.

<table>
<thead>
<tr>
<th>Network Object</th>
<th>Notes</th>
<th>Estimated Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Public IPs</td>
<td>One per deployment, assigned to Ops Manager.</td>
<td>1</td>
</tr>
<tr>
<td>Virtual Private Network (VPC)</td>
<td>One per deployment. A PCF deployment exists within a single VPC and a single AWS region, but should distribute PCF jobs and instances across 3 AWS AZs to ensure a high degree of availability.</td>
<td>1</td>
</tr>
</tbody>
</table>
| Subnets | The reference architecture requires the following subnets:  
- 1 x (/24) infrastructure (BOSH Director) subnet  
- 3 x (/24) public subnets (Ops Manager, Elastic Load Balancers, NAT instances), one per AZ  
- 3 x (/20) PAS subnets (Gorouters, Diego Cells, Cloud Controllers, etc.), one per AZ  
- 3 x (/20) services subnets (RabbitMQ, MySQL, Spring Cloud Services, etc.), one per AZ  
- 3 x (/24) RDS subnets (Cloud Controller DB, UAA DB, etc.), one per AZ. | 13 |
| This reference architecture requires 4 route tables: one for the public subnet, and one each for all 3 private subnets across 3 |
Route Tables

AZs. Consult the following list:

- **PublicSubnetRouteTable**: This routing table enables the ingress/egress routes from/to Internet through the Internet gateway for Ops Manager and the NAT Gateway.
- **PrivateSubnetRouteTable**: This routing table enables the egress routing to the Internet through the NAT Gateway for the BOSH Director and PAS.

For more information, see the Terraform [script](#) that creates the route tables and the [script](#) that performs the route table association.

Note: If an EC2 instance sits on a subnet with an Internet gateway attached as well as a public IP address, it is accessible from the Internet through the public IP address; for example, Ops Manager. PAS needs Internet access due to the access needs of using an S3 bucket as a blobstore.

The reference architecture requires 5 Security Groups. For more information, see the Terraform Security Group rules [script](#). The following table describes the Security Group ingress rules:

<table>
<thead>
<tr>
<th>Security Group</th>
<th>Port</th>
<th>From CIDR</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpsMgrSG</td>
<td>22</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>Ops Manager SSH access</td>
</tr>
<tr>
<td>OpsMgrSG</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>Ops Manager HTTP access</td>
</tr>
<tr>
<td>VmsSG</td>
<td>ALL</td>
<td>VPC_CIDR</td>
<td></td>
<td>Open up connections among BOSH-deployed VMs</td>
</tr>
<tr>
<td>MysqlSG</td>
<td>3306</td>
<td>VPC_CIDR</td>
<td>TCP</td>
<td>Enable network access to RDS</td>
</tr>
<tr>
<td>ElbSG</td>
<td>80</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>HTTP to PAS</td>
</tr>
<tr>
<td>ElbSG</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>HTTPS to PAS</td>
</tr>
<tr>
<td>ElbsSg</td>
<td>4443</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>WebSocket connection to Loggregator endpoint</td>
</tr>
<tr>
<td>SshElbSg</td>
<td>2222</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>SSH connection to containers</td>
</tr>
</tbody>
</table>

PCF on AWS requires the Elastic Load Balancer, which can be configured with multiple listeners to forward HTTP/HTTPS/TCP traffic. Two Elastic Load Balancers are recommended: one to forward the traffic to the Gorouters, [PcfElb](#), the other to forward the traffic to the Diego Brain SSH proxy, [PcfSshElb](#). For more information, see the Terraform load balancers [script](#).

The following table describes the required listeners for each load balancer:

<table>
<thead>
<tr>
<th>ELB</th>
<th>Instance/Port</th>
<th>LB Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PcfElb</td>
<td>gorouter/80</td>
<td>80</td>
<td>HTTP</td>
<td>Forward traffic to Gorouters</td>
</tr>
<tr>
<td>PcfElb</td>
<td>gorouter/80</td>
<td>443</td>
<td>HTTPS</td>
<td>SSL termination and forward traffic to Gorouters</td>
</tr>
<tr>
<td>PcfElb</td>
<td>gorouter/80</td>
<td>4443</td>
<td>SSL</td>
<td>SSL termination and forward traffic to Gorouters</td>
</tr>
<tr>
<td>PcfSshElb</td>
<td>diego-brain/2222</td>
<td>2222</td>
<td>TCP</td>
<td>Forward traffic to Diego Brain for container SSH connections</td>
</tr>
</tbody>
</table>

Each ELB binds with a health check to check the health of the back-end instances:

- [PcfElb](#) checks the health on Gorouter port 80 with TCP
- [PcfSshElb](#) checks the health on Diego Brain port 2222 with TCP

Jumpbox

Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access to Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaasSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.

Integrate PCF with Customer Data Center through VPN

At times, applications on PCF need to access on-premise data. The connection between an AWS VPC and an on-premise datacenter is made through [VPN peering](#). When employing non-VPN peering, there are several points to consider:

1. Assign routable IP addresses with the following in mind:
- It may not be realistic to request multiple routable /22 address spaces, due to IP exhaustion.
- Using different VPC address spaces can cause snowflakes deployments and present difficulties in automation.
- Only make the load balancer, NAT devices, and Ops Manager routable.
- PCF components can route egress through a NAT instance. As a result, operators do not need to assign routable IP addresses to PCF components.

2. Inbound traffic from the datacenter should come through an internal load balancer.

3. Outbound traffic to the datacenter should go through AWS NAT instances.
Reference Architecture for Pivotal Cloud Foundry on Azure

Page last updated:

This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on Azure.

Azure does not provide resources in a way that translates directly to PCF availability zones. Instead, Azure provides high availability through fault domains and availability sets.

All reference architectures described in this topic are validated for production-grade PCF deployments using fault domains and availability sets that include multiple job instances.

See Azure on PCF Requirements for general requirements for running PCF and specific requirements for running PCF on Azure.

PCF Reference Architectures

A PCF reference architecture describes a proven approach for deploying PCF on a specific IaaS, such as Azure, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base Azure Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on Azure.

Base Reference Architecture Components
The following table lists the components that are part of a base reference architecture deployment on Azure using a single resource group.

<table>
<thead>
<tr>
<th>Component</th>
<th>Reference Architecture Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domains and DNS</td>
<td>CF Domain Zones and routes in use by the reference architecture include: domains for &quot;.apps&quot; and &quot;.system&quot; (required), a route for Ops Manager (required), a route for Doppler (required), a route for Loggregator (required), a route for SSH access to app containers (optional), and a route for TCP routing to apps (optional).</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Deployed on the management subnet and accessible by FQDN or through an optional jumpbox.</td>
</tr>
<tr>
<td>BOSH</td>
<td>Deployed on the management subnet.</td>
</tr>
<tr>
<td>Azure Load Balancer - API and Apps</td>
<td>Required. Load balancer that handles incoming API and apps requests and forwards them to the Gorouters.</td>
</tr>
<tr>
<td>Azure Load Balancer - ssh-proxy</td>
<td>Optional. Load balancer that provides SSH access to app containers.</td>
</tr>
<tr>
<td>Azure Load Balancer - tcp-router</td>
<td>Optional. Load balancer that handles TCP routing requests for apps.</td>
</tr>
<tr>
<td>Azure Load Balancer - MySQL</td>
<td>Required to provide high availability for MySQL backend to Pivotal Application Service (PAS).</td>
</tr>
<tr>
<td>Gorouters</td>
<td>Accessed through the API and Apps load balancer. Deployed on the PAS subnet, one job per Azure availability set.</td>
</tr>
<tr>
<td>Diego Brains</td>
<td>Required. However, the SSH container access functionality is optional and enabled through the SSH Proxy load balancer. Deployed on the PAS subnet, one job per Azure availability set.</td>
</tr>
<tr>
<td>TCP Routers</td>
<td>Optional feature for TCP routing. Deployed on the PAS subnet, one job per availability zone.</td>
</tr>
<tr>
<td>MySQL</td>
<td>Reference architecture uses internal MySQL provided with PCF. Deployed on the PAS subnet, one job per Azure availability set.</td>
</tr>
<tr>
<td>PAS</td>
<td>Required. Deployed on the PAS subnet, one job per Azure availability set.</td>
</tr>
<tr>
<td>Storage Accounts</td>
<td>PCF on Azure requires 5 standard storage accounts: BOSH, Ops Manager, and three PAS storage accounts. Each account comes with a set amount of disk. Reference architecture recommends using 5 storage accounts because Azure Storage Accounts have an IOPs limit of approximately 20k per account, which generally relates to a BOSH JOB/VM limit of approximately 20 VMs each.</td>
</tr>
<tr>
<td>Service Tiles</td>
<td>Deployed on the Services subnet. Each service tile is deployed to an availability set.</td>
</tr>
</tbody>
</table>

**Alternative Network Layouts for Azure**

This section describes the possible network layouts for PCF deployments as covered by the reference architecture of PCF on Azure.

At a high level, there are currently two possible ways of deploying PCF as described by the reference architecture:

1. Single resource group, or
2. Multiple resource groups.

The first scenario is outlined in [Installing PCF on Azure](#). It models a single PCF deployment in a single Azure Resource Group.

If you require multiple resource groups, refer to the [Multiple Resource Group deployment](#) section of this topic.

**Network Layout**
This diagram illustrates the network topology of the base reference architecture for PCF on Azure. In this deployment, you expose only a minimal number of public IP addresses and deploy only one resource group.

Network Objects

The following table lists the network objects in PCF on Azure reference architecture.

<table>
<thead>
<tr>
<th>Network Object</th>
<th>Notes</th>
<th>Estimated Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Public IP addresses</td>
<td>Use 1. global IP address for apps and system access 2. Ops Manager or optional jumpbox. Optionally, you can use a public IP address for the ssh-proxy and tcp-router load balancers.</td>
<td>1-4</td>
</tr>
<tr>
<td>Virtual Network</td>
<td>One per deployment. Azure virtual network objects allow multiple subnets with multiple CIDRs, so a typical deployment of PCF will likely only ever require one Azure Virtual Network object.</td>
<td>1</td>
</tr>
<tr>
<td>Subnets</td>
<td>Separate subnets for 1. management (Ops Manager, BOSH Director, Jumpbox), 2. PAS, 3. and services. Using separate subnets allows you to configure different firewall rules due to your needs.</td>
<td>4</td>
</tr>
<tr>
<td>Routes</td>
<td>Routes are typically created by Azure dynamically when subnets are created, but you may need to create additional routes to force outbound communication to dedicated SNAT nodes. These objects are required to deploy PCF without public IP addresses.</td>
<td>3+</td>
</tr>
<tr>
<td>Firewall Rules</td>
<td>Azure firewall rules are collected into a Network Security Group (NSG) and bound to a Virtual Network object and can be created to use IP ranges, subnets, or instance tags to match for source and destination fields in a rule. One NSG can be used for all firewall rules.</td>
<td>12</td>
</tr>
<tr>
<td>Load Balancers</td>
<td>Used to handle requests to Gorouters and infrastructure components. Azure uses 1 or more load balancers. The API and Apps load balancer is required. The TCP Router load balancer used for TCP routing feature and the SSH load balancer that allows SSH access to Diego apps are both optional. In addition, you can use a MySQL load balancer to provide high availability to</td>
<td>1-4</td>
</tr>
</tbody>
</table>
MySQL. This is also optional.

Jumpbox
Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaaSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.

Multiple Resource Group Deployment

This diagram illustrates the case where you want to use additional resource groups in your PCF deployment on Azure.

Shared network resources may already exist in an Azure subscription. In this type of deployment, using multiple resource groups allows you to reuse existing resources instead of provisioning new ones.

To use multiple resource groups, you need to provide the BOSH Service Principal with access to the existing network resources.

Multiple Resource Groups Deployment Notes

To deploy PCF on Azure with multiple resource groups, you can define custom roles to grant resource group access to your BOSH Service Principal. For example, you might develop the following:

- Dedicated Network Resource Group, limits BOSH Service Principal so that it does not have admin access to network objects.
- Custom Role for BOSH Service Principal, applied to Network Resource Group, limits the BOSH Service Principal to minimum read-only access.
The actions prefixed with `Microsoft.Network/publicIPAddresses` are only required if using IP addresses.

- Custom Role for BOSH Service Principal, applied to Subscription, allowing the Operator to deploy PCF components

```json
az role definition create --role-definition
{
  "Name": "PCF Deploy Min Perms",
  "IsCustom": true,
  "Description": "MVP PCF Terraform Perms",
  "Actions": [{
    "Microsoft.Compute/register/action"
  },
  "NotActions": [],
  "AssignableScopes": ["/subscriptions/[YOUR_SUBSCRIPTION_ID]"
}
}
```

- Custom roles can be assigned to service principals with `az role assignment create --role ROLE --assignee SERVICE_PRINCIPAL_ID`.

---

**Multiple Resource Group and Multiple Load Balancer Deployment**

This diagram illustrates the case where you want to deploy multiple load balancers with your multiple resource group deployment.

The key design points of this deployment are the following:

- Two Azure load balancer (ALBs) to the Gorouter exist. The first ALB is for API access, which utilizes a private IP address. The system domain should resolve to this ALB. The second ALB is for application access, which can either use a public or private IP address. The apps domain should resolve to this ALB.
- Azure does not allow the Gorouters to be members of more than one ALB member pool for the same ports, for example `80` and `443`. This restriction requires an additional reverse proxy to front the Gorouters to allow them to expose traffic on these ports for the system domain.
Log Analytics Integration

Azure Log Analytics is a service that helps you collect and analyze data generated by resources in your cloud and on-premises environments. The Microsoft Azure Log Analytics Nozzle for PCF receives logs and metrics from the Loggregator Firehose, filters and resolves the events, and then sends them to Log Analytics, where they appear on unified dashboards and can be correlated with and alerted on other Azure resources.

DNS Delegation

Azure DNS supports DNS delegation, allowing for sub-level domains to be hosted within Azure. This functionality is fully supported within PCF.

Pivotal recommends that use a sub-zone for your PCF deployment. For example, if your company’s domain is example.com, your PCF zone in Azure DNS would be pcf.example.com.

As Azure DNS does not support recursion, in order to properly configure Azure DNS, create an NS record with your registrar which points to the four name servers supplied by your Azure DNS Zone configuration. Once your NS records have been created, you can then create the required wildcard A records for the PCF application and system domains, as well as any other records desired for your PCF deployments.

You do not need to make any configuration changes in PCF to support Azure DNS.

Azure Blob Storage

Due to limitations in PCF, it is not possible to support a high-availability deployment of the backing NFS store needed for droplets, buildpacks, and other resources. Azure Blob Storage provides fully-redundant hot, cold, or archival storage in either local, regional, or global offerings. It is recommended to use Azure Blob Storage as the external File Storage to provide unlimited scaling and redundancy for high-availability deployments of PCF.

To enable Azure Blob Storage, do the following:

1. Create a Storage Account with the level of redundancy you require. The storage account does not need to be in the same Resource Group as PCF.
2. Create Containers for the buildpacks, droplets, packages, and resources required by PCF.
3. Open Ops Manager.

4. Select Pivotal Application Services.

5. Click the Settings Tab.


7. Click External Azure Account.

8. Enter the Storage Account details.

9. In the Ops Manager main page, apply the changes when you are ready to reconfigure.

Note: There is no direct path to migrate previous objects to Azure Blob Storage. Contact Pivotal Support if you need assistance with this migration.

Load Balancer Migrations

The Azure Load Balancer has two SKUs: Standard and Basic. The Standard SKU Load Balancer is a new load balancer for all TCP and UDP applications with an expanded and more granular feature set over the Basic Load Balancer. Azure’s Standard SKU Load Balancer lets you scale your applications and create high availability in environments ranging from small scale deployments to large and complex multi-zone architectures. While the Basic SKU Load Balancer works within the scope of an availability set, a Standard Load Balancer covers an entire virtual network. Pivotal recommends using the Standard SKU Load Balancer instead of the Basic SKU Load Balancer. If you currently use the Basic SKU Load Balancer and wish to migrate to the the Standard SKU, please see the Migrate Basic SKU Load Balancer to Standard SKU Load Balancer documentation on GitHub.
Reference Architecture for Pivotal Cloud Foundry on GCP

This topic presents two reference architectures for installing Pivotal Cloud Foundry (PCF) on Google Cloud Platform (GCP): on a shared virtual private cloud (VPC) and on a single-project VPC. This topic also outlines multiple networking variants for VPC deployment. The architectures are validated for production-grade PCF deployments using multiple AZs.

See [PCF on GCP Requirements](#) for general requirements for running PCF and specific requirements for running PCF on GCP.

PCF Reference Architectures

A PCF reference architecture describes a proven approach for deploying PCF on a specific IaaS, such as GCP.

A PCF reference architecture must meet the following requirements:

- Be secure
- Be publically-accessible
- Include common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Be able to host at least 100 app instances

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Shared vs Single-Project VPCs

A shared VPC installation is harder to configure than a PCF deployment on a single-project VPC, because the required account privileges and resource allocations are more granular and complex. But the shared VPC architecture allows network assets to be centrally located, which simplifies auditing and security. Pivotal recommends the shared VPC model for:

- Deployments with deep auditing and security requirements
- When networks hosting the foundation need to connect back to an internal network via VPN or interconnect

A single-project VPC lets the platform architect give PCF full access to the VPC and its resources, which makes configuration easier. Pivotal recommends single-project VPC architecture for:

- Standalone deployments that do not connect to an internal network
- Test and experimental deployments, and for projects which do not belong to an organization

Shared VPC GCP Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on a shared VPC on GCP. This architecture requires an organization on the VPC that contains a host project and a service project.

For more information about shared VPCs on GCP, see [Shared VPC Overview](#) in the Google Cloud documentation.

For more information about how this architecture divides resources between projects, see [Host/Service Architecture](#).
NAT Network Topology

To expose a minimal number of public IP addresses, set up your NAT as shown in this diagram.
Cloud Interconnect

To speed communication between data centers, use Google Cloud Interconnect as shown in this diagram.

View a larger version of this diagram.
Host / Service Architecture

GCP allocates resources using a hierarchy that centers around projects. To create a VPC, architects define a host project that allocates network resources for the VPC, such as address space and firewall rules. Then they can define one or more service projects to run within the VPC, which share the network resources allocated by the host project and include their own non-network resources, such as VMs and storage buckets.

To install PCF in a shared VPC on GCP, you create a host project for the VPC and a service project dedicated to running PCF.

Host Project Resources

The host project centrally manages the following shared VPC network resources for PCF:

- Infra subnet (Ops Manager and BOSH Director)
- PAS subnet
- Services subnet
- Isolation Segments
- Firewall rules
- NAT instances and gateway
- VPN/interconnect
- Routes, e.g. egress internet through NAT or egress on-premises through VPN

Service Project Resources

The PCF service project manages the following resources:

- Google Cloud Compute instances (VMs)
  - BOSH
  - Ops Manager
  - VMs deployed by BOSH, such as PCF and service components
- Google Cloud Storage buckets, for blobstore
  - BOSH Director
  - Resources
  - Buildpacks
  - Droplets
  - Packages
- Service account and a service account key for PCF to access the storage buckets
- A service account for PCF
- Load Balancers
- Google Cloud SQL instances, if using external databases

Single-Project VPC Base GCP Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on a single-project VPC on GCP.
Base Reference Architecture Components

The following table lists the components that are part of a reference architecture deployment with three availability zones.

<table>
<thead>
<tr>
<th>Component</th>
<th>Reference Architecture Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domains &amp; DNS</td>
<td>CF Domain Zones and routes in use by the reference architecture include: domains for *.apps and *.system (required), a route for Ops Manager (required), a route for doppler (required), a route for Loggregator (required), a route for ssh access to app containers (optional) and a route for TCP routing to apps (optional). Reference architecture uses GCP Cloud DNS as the DNS provider.</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Deployed on the infrastructure subnet and accessible by FQDN or through an optional jumpbox.</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>Deployed on the infrastructure subnet.</td>
</tr>
<tr>
<td>Gorouters</td>
<td>Accessed through the HTTP and TCP WebSockets load balancers. Deployed on the Pivotal Application Service (PAS) subnet, one job per availability zone.</td>
</tr>
<tr>
<td>Diego Brains</td>
<td>Required. However, the SSH container access functionality is optional and enabled through the SSH Proxy load balancer. Deployed on the PAS subnet, one job per availability zone.</td>
</tr>
<tr>
<td>TCP Routers</td>
<td>Optional feature for TCP routing. Deployed on the PAS subnet, one job per availability zone.</td>
</tr>
<tr>
<td>CF Database</td>
<td>Reference architecture uses GCP Cloud SQL rather than internal databases. Configure your database with a strong password and limit access only to components that require database access.</td>
</tr>
<tr>
<td>CF Blob Storage and Buckets</td>
<td>For buildpacks, droplets, packages and resources. Reference architecture uses Google Cloud Storage rather than internal file storage.</td>
</tr>
<tr>
<td>Services</td>
<td>Deployed on the PCF managed services subnet. Each service is deployed to each availability zone.</td>
</tr>
</tbody>
</table>

Alternative GCP Network Layouts for PCF

This section describes the possible network layouts for PCF deployments as covered by the reference architecture of PCF on GCP.

At a high level, there are currently two possible ways of granting public Internet access to PCF as described by the reference architecture:
- NAT provides connectivity from PCF internals to the public Internet.
  - The instructions for Installing PCF on GCP Manually use this method.
- Every PCF VM receives its own public IP address (no NAT).
  - The instructions for Installing PCF on GCP using Terraform use this method.

However, if you require NAT, you may refer to the following section.

### NAT-based Solution

This diagram illustrates the case where you want to expose only a minimal number of public IP addresses.

![NAT-based Solution Diagram](image)

View a larger version of this diagram.

### Public IP addresses Solution

If you prefer not to use a NAT solution, you can configure PCF on GCP to assign public IP addresses for all components. This type of deployment may be more performant since most of the network traffic between Cloud Foundry components are routed through the front end load balancer and the Gorouter.

### Network Objects

The following table lists the network objects expected for each type of reference architecture deployment with three availability zones (assumes you are using NAT).

<table>
<thead>
<tr>
<th>Network Object</th>
<th>Notes</th>
<th>Minimum Number: NAT-based</th>
<th>Minimum Number: Public IP Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>External IPs</td>
<td>For a NAT solution, use global IP address for apps and system access, and Ops Manager or optional jumpbox</td>
<td>2</td>
<td>30+</td>
</tr>
<tr>
<td>NAT</td>
<td>One NAT per availability zone.</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Network</td>
<td>One per deployment. GCP Network objects allow multiple subnets with multiple CIDRs, so a typical deployment of PCF will likely only ever require one GCP Network object</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Separate subnets for infrastructure (Ops Manager, BOSH Director, Jumpbox), PAS, and services. Using separate
Network Communication in GCP Deployments

This section provides more background on the reasons behind certain network configuration decisions, specifically for the Gorouter.

Load Balancer to Gorouter Communications and TLS Termination

In a PCF on GCP deployment, the Gorouter receives two types of traffic:

1. Unencrypted HTTP traffic on port 80 that is decrypted by the HTTP(S) load balancer.
2. Encrypted secure web socket traffic on port 443 that is passed through the TCP WebSockets load balancer.

TLS is terminated for HTTPS on the HTTP load balancer and is terminated for WebSockets (WSS) traffic on the Gorouter.

PCF deployments on GCP use two load balancers to handle Gorouter traffic because HTTP load balancers currently do not support WebSockets.

ICMP

GCP routers do not respond to ICMP; therefore, Pivotal recommends disabling ICMP checks in BOSH Director network configuration.
Reference Architecture for Pivotal Cloud Foundry on OpenStack

This guide presents a reference architecture for Pivotal Cloud Foundry (PCF) on OpenStack. This architecture is valid for most production-grade PCF deployments in a single project using three availability zones (AZs).

See [OpenStack on PCF Requirements](#) for general requirements for running PCF and specific requirements for running PCF on OpenStack.

PCF Reference Architectures

A PCF reference architecture describes a proven approach for deploying Pivotal Cloud Foundry on a specific IaaS, such as OpenStack, that meets the following requirements:

- Secure
- Publicly-accessible
- Includes common PCF-managed services such as MySQL, RabbitMQ, and Spring Cloud Services
- Can host at least 100 app instances, or far more

Pivotal provides reference architectures to help you determine the best configuration for your PCF deployment.

Base OpenStack Reference Architecture

The following diagram provides an overview of a reference architecture deployment of PCF on OpenStack using three AZs.

Base Reference Architecture Components

The following table lists the components that are part of a base reference architecture deployment on OpenStack with three AZs.
Domains & DNS

CF Domain zones and routes in use by the reference architecture include:

- zones for *.apps and *.sys (required)
- a route for Ops Manager (required)
- a route for Doppler (required)
- a route for Loggregator (required)
- a route for ssh access to app containers (optional)
- a route for tcp access to tcp routers (optional)

Ops Manager
Deployed on the infrastructure network and accessible by FQDN or through an optional jumpbox.

BOSH Director
Deployed on the infrastructure network.

Application Load Balancer
Required. Load balancer that handles incoming HTTP, HTTPS, TCP, and SSL traffic and forwards them to the Gorouters. Load balancers are outside the scope of this document.

SSH Load Balancer
Optional. Load balancer that provides SSH access to application containers for developers. Load balancers are outside the scope of this document.

Gorouters
Accessed through the Application Load Balancer. Deployed on the Pivotal Application Service (PAS) network, one per AZ.

Diego Brains
This component is required. However, the SSH container access functionality is optional and enabled through the SSH Load Balancers. Deployed on the PAS network, one per AZ.

TCP Routers
Optional feature for TCP routing. Deployed on the PAS network, one per AZ.

CF Database
Reference architecture uses internal MySQL.

Storage Buckets
Reference architecture uses customer provided blobstore. Buckets are needed for BOSH and PAS.

Service Balancers
Deployed on the services network.

Gorouters
Accessed through the Application Load Balancer. Deployed on the Pivotal Application Service (PAS) network, one per AZ.

Diego Brains
This component is required. However, the SSH container access functionality is optional and enabled through the SSH Load Balancers. Deployed on the PAS network, one per AZ.

TCP Routers
Optional feature for TCP routing. Deployed on the PAS network, one per AZ.

CF Database
Reference architecture uses internal MySQL.

Service Accounts
Two service accounts are recommended: one for OpenStack “paving,” and the other for Ops Manager and BOSH. Consult the following list:

- Admin Account: Concourse will use this account to provision required OpenStack resources as well as a Keystone service account.
- Keystone Service Account: This service account will be automatically provisioned with restricted access only to resources needed by PCF.

OpenStack Quota
The default compute quota on a new OpenStack subscription is typically not enough to host a multi-AZ deployment. The recommended quota for instances is 100. Your OpenStack network quotas may also need to be increased.

OpenStack Objects

The following table lists the network objects in this reference architecture.

<table>
<thead>
<tr>
<th>Network Object</th>
<th>Notes</th>
<th>Estimated Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating IP addresses</td>
<td>Two per deployment. One assigned to Ops Manager, the other to your jumpbox.</td>
<td>2</td>
</tr>
<tr>
<td>Project</td>
<td>One per deployment. A PCF deployment exists within a single project and a single OpenStack region, but should distribute PCF jobs and instances across three OpenStack AZs to ensure a high degree of availability.</td>
<td>1</td>
</tr>
</tbody>
</table>
| Networks | The reference architecture requires the following Tenant Networks:  
- 1 x (/24) Infrastructure (Ops Manager, BOSH Director, Jumpbox).  
- 1 x (/20) PAS (Gorouters, Diego Cells, Cloud Controllers, etc.).  
- 1 x (/20) Services (RabbitMQ, MySQL, Spring Cloud Services, etc.)  
- 1 x (/24) On-demand services (Various.)  
An Internet-facing network is also required:  
- 1 x Public network. | 5 |

Note: In many cases, the public network is an “under the cloud” network that is shared across projects.
**Routers**

This reference architecture requires one router attached to all networks:
- **VirtualRouter**: This router table enables the ingress/egress routes from/to Internet to the project networks and provides sNAT services.

**Security Groups**

The reference architecture requires one Security Groups. The following table describes the Security Group ingress rules:

<table>
<thead>
<tr>
<th>Security Group</th>
<th>Port</th>
<th>From CIDR</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpsMgrSG</td>
<td>22</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>Ops Manager SSH access</td>
</tr>
<tr>
<td>OpsMgrSG</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>Ops Manager HTTP access</td>
</tr>
<tr>
<td>VmsSG</td>
<td>ALL</td>
<td>VPC_CIDR</td>
<td>ALL</td>
<td>Open up connections among BOSH-deployed VMs</td>
</tr>
</tbody>
</table>

Additional security groups may be needed which are specific to your chosen load balancing solution.

**Load Balancers**

PCF on OpenStack requires a load balancer, which can be configured with multiple listeners to forward HTTP/HTTPS/TCP traffic. Two load balancers are recommended: one to forward the traffic to the Gorouters, `AppsLB`, the other to forward the traffic to the Diego Brain SSH proxy, `SSHLB`.

The following table describes the required listeners for each load balancer:

<table>
<thead>
<tr>
<th>Name</th>
<th>Instance/Port</th>
<th>LB Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppsLB</td>
<td>gorouter/80</td>
<td>80</td>
<td>HTTP</td>
<td>Forward traffic to Gorouters</td>
</tr>
<tr>
<td>AppsLB</td>
<td>gorouter/80</td>
<td>443</td>
<td>HTTPS</td>
<td>SSL termination and forward traffic to Gorouters</td>
</tr>
<tr>
<td>SSHLB</td>
<td>diego-brain/2222</td>
<td>2222</td>
<td>TCP</td>
<td>Forward traffic to Diego Brain for container SSH connections</td>
</tr>
</tbody>
</table>

Each load balancer needs a check to validate the health of the back-end instances:
- `AppsLB` checks the health on Gorouter port 80 with TCP
- `SSHLB` checks the health on Diego Brain port 2222 with TCP

**Note**: In many cases, the load balancers are provided as an “under the cloud” service that is shared across projects.

**Jumpbox**

Optional. Provides a way of accessing different network components. For example, you can configure it with your own permissions and then set it up to access to Pivotal Network to download tiles. Using a jumpbox is particularly useful in IaaSes where Ops Manager does not have a public IP address. In these cases, you can SSH into Ops Manager or any other component through the jumpbox.
vSphere Reference Architecture

Overview

Pivotal validates the reference architectures described in this topic against multiple production-grade usage scenarios.

This document does not replace the installation instructions provided in the PCF on vSphere Requirements topic. Instead, it gives examples of how to apply those instructions to real-world production environments.

<table>
<thead>
<tr>
<th>PCF Products Validated</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>vSphere</td>
<td>6.5</td>
</tr>
<tr>
<td>VMware NSX-T</td>
<td>2.2</td>
</tr>
<tr>
<td>Pivotal Cloud Foundry Operations Manager</td>
<td>2.2 or later</td>
</tr>
<tr>
<td>PAS</td>
<td>2.2 or later</td>
</tr>
<tr>
<td>PKS</td>
<td>1.1.5 or later</td>
</tr>
</tbody>
</table>

Base vSphere Reference Architecture

This reference architecture includes VMware vSphere and NSX-T, a software-defined network virtualization platform that runs on VMware ESXi virtual hosts.

The reference architecture supports capacity growth at the vSphere and PCF levels as well as installation security through the NSX-T firewall. It allocates a minimum of three servers to each vSphere cluster and spreads PCF components across three clusters for high availability (HA). Each vSphere cluster corresponds to an Availability Zone (AZ). For information about HA in PCF, see High Availability in Cloud Foundry.

> **Note:** Small and non-critical deployments can maintain high availability with two vSphere clusters instead of three, provided that the PAS system databases are configured as external. For more information, see Configuring PAS.

To use all features listed in this topic for your PAS installation, you must have Advanced or above licensing from VMware for NSX-T. For PKS, the required licensing is included by default.

If you want to deploy PCF without NSX-T, see vSphere Reference Architecture, which includes NSX-V design considerations.

Non-SDN Deployments

This section provides information about VLAN and container-to-container networking considerations for PCF deployments that do not use software-defined networking (SDN).

VLAN Considerations

Without an SDN environment, you draw network address space and broadcast domains from the greater data center pool. For this approach, you need to deploy PAS and PKS in a particular alignment.
When designing a non-SDN environment, ensure that VLANs and address space for PAS and PKS do not overlap. In addition, note that non-SDN PCF environments have the following characteristics:

- Load balancing is performed external to the PCF installation.
- Client SSL termination must happen either at the load balancer or Gorouters, or both. Ensure your certificates can accomplish this.
- Firewall layer is accomplished external to the PCF installation.

### Container-to-Container Networking and CNI Considerations

Changing your container-to-container networking strategy after deployment is possible. However, with NSX-T, you need both the VMware NSX-T Container Plug-In for PCF tile and NSX-T infrastructure. For information about deploying PAS on vSphere with NSX-T internal networking using the VMware NSX-T Container Plug-In for PCF tile, see [Deploying PAS with NSX-T Networking](#).

Migrating from a non-SDN environment to an SDN-enabled solution is possible but best considered as a greenfield deployment. Inserting an SDN layer under an active PCF installation is disruptive.

### PAS Without SDN

You can use PAS without an SDN overlay. For more information, see [PCF on vSphere Requirements](#).

### PKS Without SDN

NSX-T SDN is included in PKS by default, along with the associated licensing. Alternatively, you can use a built-in network stack, Flannel, which handles container networking.

If you want to deploy PKS without NSX-T, select Flannel as your container networking interface in the Networking pane of the PKS tile. For information
about configuring your container networking interface, see the Networking section in Installing PKS on vSphere.

In addition, you must define networks for deploying PKS and PKS-provisioned Kubernetes clusters. For information about defining these networks in Ops Manager, see the Create Networks Page section in Configuring Ops Manager on vSphere.

SDN-Enabled Deployments With NSX-T

This section provides information about network requirements, routing, and load balancing for PAS and PKS deployments that use SDN.

Network Requirements for PAS

PAS requires a number of statically defined networks to host the main components it is composed of. For the Tenant side of an NSX-T deployment, a series of non-routable address banks that the NSX-T routers will manage is defined as follows:

- **Infrastructure**: 192.168.1.0/24
- **Deployment**: 192.168.2.0/24
- **Services**: 192.168.3.0/24

  The Services network can be used with other Ops Manager tiles that you install in addition to PAS. Some of these tiles may require on-demand network capacity. Pivotal recommends that you consider adding a network per tile that needs on-demand resources and pair them up in the tile’s configuration. For more information, see OD-Services# below.

  - **OD-Services#**: 192.168.4.0 - 192.168.9.0 in /24 segments

    For example, the Redis for PCF tile asks for Network and Services Network. The first one is for placing the broker, and the second one is for deploying worker VMs to support the service. In this scenario, you can deploy a new network OD-Services1 and instruct Redis for PCF to use the Services network for the broker and the OD-Services1 network for the workers. The next tile can use the Services network for the broker and a new OD-Services2 network for workers and so on.

- **Isolation Segments**: 192.168.10.0 - 192.168.63.0 in /24 segments

  Isolation segments can be added to an existing PAS installation. This range of address space is used when you add one or more segments. A /24-network in this range should be deployed for each new isolation segment. The capacity of this address bank is sufficient for more than 50 isolation segments, each having more than 250 VMs.

- **PAS Dynamic Orgs**: 192.168.128.0/17 = 128 orgs/foundation

  Dynamically assigned Org networks are attached to automatically generated NSX-T Tier-1 routers. Instead of defining these networks in Ops Manager, the operator provides a non-overlapping block of address space. This is configurable in the NCP pane of the VMware NSX-T Container Plug-In for PCF tile in Ops Manager. Every Org receives a new /24 network.

  This reference uses a pattern that follows previous references. However, all networks now break on the /24 boundary, and the network octet is numerically sequential (1-2-3).

Network Requirements for PKS

When deploying PKS through Ops Manager, you must allocate a block of address space for dynamic networks that PKS will deploy per namespace.

Network requirements for PKS are as follows:

- **PKS Clusters**: 172.24.0.0/14
- **PKS Pods**: 172.28.0.0/14

Summary

The complete addressing strategy for both PAS and PKS is as follows:

- **Infrastructure**: 192.168.1.0/24
- **Deployment**: 192.168.2.0/24
- **Services**: 192.168.3.0/24
- **OD-Services#**: 192.168.4.0 - 192.168.9.0 in /24 segments

© Copyright Pivotal Software Inc, 2013-2019 564
Isolation Segments: 192.168.10.0 - 192.168.63.0 in /24 segments
Undefined: 192.168.64.0 - 192.168.127.0
PAS Dynamic Orgs: 192.168.128.0/17
PKS Clusters: 172.24.0.0/14
PKS Pods: 172.28.0.0/14

Network Requirements for External Routing

Routable external IPs on the provider side, for example, for NATs, PAS orgs, and load balancers, are assigned to the T0 router, which is located in front of the PCF installation. There are two approaches to assigning address space blocks to this job:

- Without PKS or with PKS Ingress: The T0 router needs some routable address space to advertise on the BGP network with its peers. Select a network range with ample address space that can be split into two logical jobs: one job is advertised as a route for traffic, and the other job is for aligning T0 DNATs and SNATs, load balancers, and other jobs. Unlike with NSX-V, SNATs consume much more address space than before.
- With PKS No Ingress: Compared to the approach above, this approach has much higher address space consumption for load balancer VIPs. Therefore, allow for 4x the address space because Kubernetes service types allocate addresses frequently.

Provider routable address space is /25 or /23 for PKS No Ingress.

NSX-T handles the routing between a T0 router and any T1 routers associated with it.

PAS with NSX-T

Expanding PAS with SDN features is best considered as a greenfield effort. Inserting an SDN layer under a working PAS installation is non-trivial and likely triggers a rebuild. NSX-T constructs that can be used by PAS include the following:

- Logical networks, encapsulated broadcast domains
- VLAN exhaustion avoidance through the use of logical networks
- Routing services and NAT/SNAT to network fence the PCF installation
- Load balancing services to pool systems such as Gorouters
- SSL termination at the load balancer
- Distributed routing and firewall services at the hypervisor
The VMware NSX-T Container Plug-In for PCF tile, which provides a container networking stack instead of the built-in Silk solution, interlocks with NSX-T already deployed on the IaaS. This tile cannot be used unless NSX-T has already been established on the IaaS. These technologies cannot be retrofitted into an established PKS or PAS installation.

To deploy PAS with NSX-T, you need to provide the following in the VMware NSX-T Container Plug-In for PCF tile:

- NSX Manager host
- Username and password
- NSX Manager CA certificate
- PAS foundation name that must match the tag added to T0 router, external NAT pool, and IP block
- Subnet prefix, which controls the size of each org and defaults to /24, that is, 254 addresses

In addition, you must select Enable SNAT for Container Networks in the tile. For more information about configuring PAS with NSX-T networking, see Deploying PAS with NSX-T Networking.

Each new job, such as an isolation segment, falls to a broadcast domain or logical switch connected to a T1 router acting as the gateway to that network. This approach provides a DNAT and SNAT control point and a firewall boundary.

The Services network is an exception. It shares a T1 router with any associated OD-Services# networks. Because these networks are considered part of the same system, inserting any NATs or firewall rules between them is not needed.

Load Balancing for PAS

Without NSX-T, you need to choose a suitable load balancer to send traffic to the Gorouters and other systems. Installations approaching production level typically use external load balancing from hardware appliance vendors or other network-layer solutions.
With NSX-T, load balancing is available in the SDN layer. These load balancers are a logical entity tied to the resources in the Edge Cluster and align to the network or networks represented by a T1 router. They function as a Layer 4 load balancer. SSL termination is available on the NSX-T load balancer. However, Pivotal recommends passing the SSL through to your Gorouters.

Common deployments of load balancing in PAS are as follows:

- HTTP/HTTPS traffic to and from Gorouters
- TCP traffic to and from TCP routers
- Diego Brain

NSX-T load balancers can support many VIPs. Consider deploying one load balancer per network (T1) and one-to-many VIPs on that load balancer per job. Edge Cluster resources are consumed for load balancing. When designing your deployment, consider how many load balancers are needed and how much capacity is available for them.

BOSH can manage the members of the server pools for the NSX-T load balancers using NS Groups.

**PKS with NSX-T**

NSX-T SDN is included in PKS by default, along with the associated licensing. To use NSX-T SDN and its dynamic constructs, you need to provide the following information when configuring PKS:

- NSX Manager host
- Username and password
- NSX Manager CA certificate
- T0 router to connect dynamically created namespace networks to
- NSX IP block from which to pull networks for pods per namespace
- NSX IP block from which to pull networks for each new cluster
- Floating IP pool ID created for externally facing IPs (NATs and VIPs)

For more information about configuring PKS with NSX-T networking, see [Installing and Configuring PKS with NSX-T Integration](#).
New T1 routers are deployed on-demand as new namespaces are added to PKS. Because these can grow rapidly, allocate a large Pod IP block such as /14 in NSX-T, and then NSX-T provisions /24 address blocks for new namespaces. Reference the Pod IP block from your PKS tile configuration.

PKS v1.1.5 and later supports multiple master nodes. The number of master nodes is defined per plan in the PKS tile. You must use an odd number of master nodes to allow etcd to form a quorum. Pivotal recommends using at least 1 master node per AZ for HA and disaster recovery.

With NSX-T SDN, networks for both PKS clusters and pods are created dynamically. Pivotal recommends using multiple clusters instead of a single cluster with multiple namespaces. Multiple clusters provide security, customization per cluster, privileged containers, failure domains, and version choice.

**Ingress Routing and Load Balancing for PKS**

This section provides information about ingress routing (Layer 7) and load balancing (Layer 4) for your PKS deployment.

**Ingress Routing**

NSX-T provides a native ingress router. Third-party options include Istio or Nginx that are running as containers in the cluster.

Wildcard DNS entries are needed for pointing at the ingress service in the style of Gorouters in PAS. Domain information for ingress is defined in the manifest of your Kubernetes deployment. See the example below.
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: music-ingress
  namespace: music1
spec:
rules:
- host: music1.pks.domain.com
  http:
    paths:
    - path: /.*
      backend:
        serviceName: music-service
        servicePort: 8080

Load Balancing

When pushing a Kubernetes deployment with type set to LoadBalancer, NSX-T automatically creates a new VIP for the deployment on the existing load balancer for that namespace.

You need to specify a listening and translation port in the service, along with a name for tagging. You also specify a protocol to use. See the example below.

apiVersion: v1
kind: Service
metadata:
  ...  
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
  name: web

Storage Design

Shared storage is a requirement for PCF. You can allocate networked storage to the host clusters following one of two common approaches: horizontal or vertical. The approach you follow should reflect how your data center arranges its storage and host blocks in its physical layout.

The following describes the vertical and horizontal approaches to configuring shared storage:

- **Vertical**: You grant each cluster its own dedicated datastores, creating a cluster-aligned storage strategy. vSphere VSAN is an example of this architecture. The vertical alignment is the preferred choice for PCF as it matches the AZ model of the PaaS. Loss of storage in a cluster constitutes loss on an AZ, which is a recoverable failure when at least two more AZs are operational.
  
  For example, with six datastores ds01 through ds06, you assign datastores ds01 and ds02 to your first cluster, ds03 and ds04 to your second cluster, and ds05 and ds06 to your third cluster. You then instruct your first PCF installation to use ds01, ds03, and ds05, and your second PCF installation to use ds02, ds04, and ds06. In this arrangement, all VMs in the same installation and cluster share a dedicated datastore.

- **Horizontal**: You grant all hosts access to all datastores and assign a subset to each installation.
  
  For example, with six datastores ds01 through ds06, you grant all nine hosts access to all six datastores. You then instruct your first PCF installation to use stores ds01 through ds03 and your second PCF installation to use ds04 through ds06.

**Warning**: If you use the horizontal storage design approach, any storage loss could affect datastores in all AZs. This could result in downtime and unrecoverable loss of data. To avoid downtime and data loss, Pivotal recommends implementing the vertical storage design approach or implementing strong redundancy design at the storage array.

Highly redundant storage systems, such as hyperconverged systems, are the optimal choice for a fully HA PaaS. In addition, hyperconverged systems scale with capacity growth and are aligned to the AZ strategy of PCF.

To improve the resiliency of your deployment, you can use separate storage for the management plane of PCF, Ops Manager and BOSH, as well as separate storage for the blobstore.

**Note**: If a datastore is part of a vSphere storage cluster using sDRS (Storage DRS), you must disable the s-vMotion feature on any datastores used by PCF. Otherwise, s-vMotion activity can rename independent disks and cause BOSH to malfunction. For more information, see [How to Migrate PCF to a New Datastore in vSphere](#).
Storage Capacity

Pivotal recommends the following capacity allocation for PAS installations:

- For production use, at least 8 TB of data storage, either as one 8-TB store or a number of smaller volumes adding up to 8 TB. Frequent development may require significantly more storage to accommodate new code and buildpacks.
- For small installations, 4-6 TB of data storage is recommended.

The primary consumer of storage is NFS or WebDAV.

Note: PCF does not currently support using vSphere storage clusters with the versions of PCF validated for this reference architecture. Datastores should be listed in the vSphere tile by their native name, not the cluster name created by vCenter for the storage cluster.

When planning storage allocation for PKS installations based on Pod and Node needs, consider the following chart.

Compute and HA Considerations

In PAS, a consolidation ratio for containers should follow a conservative 4:1 ratio of vCPUs to pCPUs. You can use a more conservative ratio, which is 2:1.

A standard Diego cell is 4x16 (XL) by default. At a 4:1 ratio, you have a 16-vCPU budget, or about 12 containers.

If you want to run more containers in a cell, scale the vCPUs accordingly. However, high core count VMs become increasingly hard to schedule on the pCPU with high physical core counts in the socket.

HA considerations include the following:

- For non-production environments, the number of AZs that PAS requires for HA depends on the location of its system databases.
  - With external databases, PAS requires at least two AZs for HA.
  - With internal system databases, PAS requires at least three AZs for HA.
- For any production environment, Pivotal recommends using at least three AZs for HA.
- PKS has no inherent HA capabilities to design for. To support PKS, design HA at the IaaS, storage, power, and access layers.

PCF achieves redundancy through the AZ construct, and the loss of an AZ is not considered catastrophic. BOSH Resurrector replaces lost VMs as needed to repair a foundation.

Foundation backup and restoration is accomplished externally by BOSH Backup and Restore (BBR). For information about BBR, see Backing Up and Restoring Pivotal Cloud Foundry.
Scaling and Capacity Management

This section provides information about scaling and capacity management for PCF.

Small PCF Installations

A small-sized PCF foundation looks as follows:

- 1 vSphere cluster/1 AZ
- 2 resource pools: 1 for NSX components and 1 for PAS or PKS
- 3 hosts minimum for vSphere HA (4 hosts for vSphere VSAN)
- Shared storage/VSAN
- 1 NSX Manager
- 3 NSX controllers
- 2 large edge VMs in a cluster

PCF: Ops Manager, the BOSH Director, and Small Footprint Runtime

This approach is intended for designing a proof of concept or development-only system. Small Footprint Runtime has no upgrade path to the standard PAS tile.

Medium PCF Installations

A medium-sized PCF foundation looks as follows:

- 2 vSphere clusters/2 AZs
- 3 resource pools: 1 for NSX components and 2 for PAS/1 for PKS
- 3 hosts minimum for vSphere HA (4 hosts for vSphere VSAN) per cluster
- Shared storage/VSAN
- 1 NSX Manager
- 3 NSX controllers
- 2 large edge VMs in a cluster

PCF: Ops Manager, the BOSH Director, and PAS/PKS

For this design, Pivotal recommends replacing Small Footprint Runtime with PAS. The second AZ doubles compute capacity and expands the NSX-T footprint.

Production-Ready PCF Installations

A production-ready PCF foundation looks as follows:

- 3 vSphere clusters/3 AZs
- 4 resource pools: 1 for NSX components and 3 for PAS/1 for PKS
- 3 hosts minimum for vSphere HA (4 hosts for vSphere VSAN)
- Shared Storage/VSAN
- 1 NSX Manager
- 3 NSX Controllers
- 3 large edge VMs in a cluster

PCF: Ops Manager, the BOSH Director, and PAS/PKS

PAS and PKS with NSX-T

A fully meshed installation of PCF v2.2 or later designed using the recommendations provided in this topic looks as follows:
Common components are the NSX T0 router and the associated T1 routers. This approach ensures that any cross-traffic between PKS and PAS apps stays within the bounds of the T0. This also provides a one-stop access point to the whole installation, which simplifies deployment automation for multiple, identical installations.

Note that each design, green PKS and orange PAS, has its own Ops Manager and BOSH Director installation. You should not share an Ops Manager and BOSH Director installation between PKS and PAS.

AZs are aligned to vSphere clusters, with resource pools as an optional organizational tool to place multiple foundations into the same capacity. You can align PKS to any AZ or cluster. Keeping PKS and PAS in completely separate AZs is not required.

You can use resource pools in vSphere clusters as AZ constructs to stack different installations of PCF. As server capacity continues to increase, the efficiency of deploying independent server clusters only for one installation is low. For example, customers are commonly deploying servers with 768-GB RAM and greater.

To allow for max capacity growth, consider using an NSX-T installation per foundation.

If you want to split the PAS and PKS installations into separate network trees, behind separate T0 routers, ensure that approach meets your needs by reviewing VMware’s recommendations for T0 to T0 routing. For more information, see Reference Design Guide for PAS and PKS with VMware NSX-T Data Center in the VMware documentation.

View a larger version of this diagram

Note: You should not share an Ops Manager and BOSH Director installation between PKS and PAS in a production environment.
Using Edge Services Gateway on VMware NSX

This cookbook provides guidance on how to configure the NSX firewall, load balancing, and NAT/SNAT services for Pivotal Cloud Foundry (PCF) on vSphere installations. These NSX-provided services take the place of an external device or the bundled HAProxy VM in PCF.

This document presents the reader with fundamental configuration options of an Edge Services Gateway (ESG) with PCF and vSphere NSX. Its purpose is not to dictate the settings required on every deployment, but instead to empower the NSX Administrator with the ability to establish a known good “base” configuration and apply specific security configurations as required.

If you are using NSX, the specific configurations described here supersede any general recommendations in the Preparing Your Firewall topic.

Assumptions

This document assumes that the reader has the level of skill required to install and configure the following products:

- VMware vSphere v5.5 or greater
- NSX v6.1.x or greater
- PCF v1.6 or greater

For detailed installation and configuration information about these products, see the following:

- vSphere Documentation
- NSX Installation and Upgrade Guide
- Reference Design: VMware NSX for vSphere (NSX) Network Virtualization Design Guide
- Pivotal Cloud Foundry Documentation

General Overview

This cookbook follows a three-step recipe to deploy PCF behind an ESG:

1. Configure Firewall
2. Configure Load Balancer
3. Configure NAT/SNAT

The ESG can scale to accommodate very large PCF deployments as needed.

This cookbook focuses on a single-site deployment and makes the following design assumptions:

- There are five non-routable networks on the tenant (inside) side of the ESG.
  - The Infra network is used to deploy Ops Manager and BOSH Director.
  - The Deployment network is used exclusively by Pivotal Application Service (PAS) to deploy Cells that host apps and related elements.
  - The CF Tiles network is used for all other deployed tiles in a PCF installation.
  - The Services network is used by BOSH Director for service tiles.
  - The Container-to-Container network is used for container to container communication in the Cells.
- There is a single service provider (outside) interface on the ESG that provides Firewall, Load Balancing and NAT/SNAT services.
- The service provider (outside) interface is connected appropriately to the network backbone of the environment, as either routed or non-routed depending on the design. This cookbook does not cover provisioning of the uplink interface.
- Routable IP addresses should be applied to the service provider (outside) interface of the ESG. Pivotal recommends that you apply 10 consecutive routable IP addresses to each ESG.
  - One reserved for NSX use (Controller to Edge I/F)
  - One for NSX Load Balancer to Gorouters
  - One for NSX Load Balancer to Diego Brains for SSH to apps
  - One routable IP address, used to access the Ops Manager frontend
  - One routable IP address, used with SNAT egress

© Copyright Pivotal Software Inc, 2013-2019
Five for future use

Pivotal recommends that operators deploy the ESGs as high availability (HA) pairs in vSphere. Also, Pivotal recommends that they be sized “large” or greater for any pre-production or production use. The deployed size of the ESG impacts its overall performance, including how many SSL tunnels it can terminate.

The ESGs have an interface in each port group used by PCF as well as a port group on the service provider (outside), often called the “transit network.” Each PCF installation has a set of port groups in a vSphere DVS to support connectivity, so that the ESG arrangement is repeated for every PCF install. It is not necessary to build a DVS for each ESG/PCF install. You do not re-use an ESG amongst PCF deployments. NSX Logical Switches (VXLAN vWires) are ideal candidates for use with this architecture.

The following diagram provides an example of port groups used with an ESG:
The following diagram illustrates container-to-container networking. The overlay addresses are wrapped and transported using the underlay deployment subnet.
Prep Step: Configure DNS and Network Prerequisites

As a prerequisite, create wildcard DNS entries for system and apps domains in PCF. Map these domains to the selected IP address on the uplink (outside) interface of the ESG in your DNS server.

The wildcard DNS \(A\) record must resolve to an IP address associated with the outside interface of the ESG for it to function as a load balancer. You can either use a single IP address to resolve both the system and apps domain, or one IP address for each.

In addition, assign the following IP addresses and address ranges within your network:

1. Assign IP Addresses to the “Uplink” (outside) interface
   - Typically you have one SNAT and three DNATs per ESG.
   - IP associated for SNAT use: All PCF internal IP addresses appear to be coming from this IP address at the ESG.
   - IP associated with Ops Manager DNAT: This IP address is the publicly routable interface for Ops Manager UI and SSH access.

2. Assign “Internal” Interface IP Address Space to the Edge Gateway.
   - 192.168.10.0/26 = PCF Deployment Network (Logical Switch or Port Group)
   - 192.168.20.0/22 = Deployment Network for PAS tile
   - 192.168.24.0/22 = CF Tiles Network for all Tiles besides PAS
   - 192.168.28.0/22 = Dynamic Services network for BOSH Director-managed service tiles.
   - 10.255.0.0/16 = Container-to-Container network for intercontainer communication.

You must update the security group and load balancer information for your PCF deployments using NSX-V on vSphere through the Ops Manager API. See Updating NSX Security Group and LoadBalancer Information for more information.

Step 1: Configure Firewall

This procedure populates the ESG internal firewall with rules to protect a PCF installation.

These rules provide granular control on what can be accessed within a PCF installation. For example, rules can be used to allow or deny another PCF installation behind a different ESG access to apps published within the installation you are protecting.

This step is not required for the installation to function properly when the firewall feature is disabled or set to “Allow All.”

To configure the ESG firewall, navigate to Edge, Manage, Firewall and set the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Destination</th>
<th>Service</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Ingress -&gt; Ops Manager</td>
<td>any</td>
<td>IP_of_OpsMgr</td>
<td>SSH, HTTP, HTTPS</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Ingress -&gt; PAS</td>
<td>any</td>
<td>IP_of_NSX-LB</td>
<td>HTTP, HTTPS</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Ingress -&gt; SSH for Apps</td>
<td>any</td>
<td>tcp:IP_of_DiegoBrain:2222</td>
<td>any</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Ingress -&gt; TCProuter</td>
<td>any</td>
<td>tcp:IP_of_NSX-TCP-LB:5000</td>
<td>any</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Inside &lt;-&gt; Inside</td>
<td>192.168.10.0/26 192.168.20.0/22 192.168.24.0/22 192.168.28.0/22</td>
<td>192.168.10.0/26 192.168.20.0/22 192.168.24.0/22 192.168.28.0/22</td>
<td>any</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Egress -&gt; IaaS</td>
<td>192.168.10.0/26</td>
<td>IP_of_vCenter IPs_of_ESXi-Svrs</td>
<td>HTTP, HTTPS</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Egress -&gt; DNS</td>
<td>192.168.0.0/16</td>
<td>IPs_of_DNS</td>
<td>DNS, DNS-UDP</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Egress -&gt; NTP</td>
<td>192.168.0.0/16</td>
<td>IPs_of_NTP</td>
<td>NTP</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Egress -&gt; SYSLOG</td>
<td>192.168.0.0/16</td>
<td>IPs_of_Syslog:514</td>
<td>SYSLOG</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow ICMP</td>
<td>192.168.10.0/26</td>
<td>*</td>
<td>ICMP</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Egress -&gt; LDAP</td>
<td>192.168.10.0/26 192.168.20.0/22</td>
<td>IPs_of_LDAP:389</td>
<td>LDAP, LDAP-over-ssl</td>
<td>Accept</td>
</tr>
<tr>
<td>Allow Egress -&gt; All Outbound</td>
<td>192.168.0.0/16</td>
<td>any</td>
<td>any</td>
<td>Accept</td>
</tr>
<tr>
<td>Default Rule</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>Deny</td>
</tr>
</tbody>
</table>
Step 2: Configure Load Balancer

The ESG provides software load balancing functionality, equivalent to the bundled HAPROXY that is included with PCF, or hardware appliances such as an F5 or A10 load balancer.

This step is required for the installation to function properly.

There are seven high level steps to this procedure:

1. Import SSL certificates to the Edge for SSL termination.
2. Enable the load balancer.
3. Create Application Profiles in the Load Balancing tab of NSX.
4. Create Application Rules in the load balancer.
5. Create Service Monitors for each pool type.
6. Create Application Pools for the multiple groups needing load balancing.
7. Create a virtual server (also known as a VIP) to pool balanced IP addresses.

What you will need:

- PEM files of SSL certificates provided by the certificate supplier for only this installation of PCF, or the self-signed SSL certificates generated during PCF installation.

In this procedure you marry the ESG’s IP address used for load balancing with a series of internal IP addresses provisioned for Gorouters in PCF. It is important to know the IP addresses used for the Gorouters beforehand.

These IP addresses can be pre-selected or reserved prior to deployment (recommended) or discovered after deployment by looking them up in BOSH Director, which lists them in the release information of the PAS installation.

Step 2.1: Import SSL Certificate

PCF requires SSL termination at the load balancer.

Note: If you intend to pass SSL termination through the load balancer directly to the Gorouters, you can skip the step below and select Enable SSL Passthru on the HTTPS Application Profile.

To enable SSL termination at the load balancer in ESG, access the ESG UI and do the following:

1. Select Edge, Manage, Settings, and then Certificates.
2. Click Green Plus button to Add Certificate.
3. Insert PEM file contents from the Networking configuration screen of PAS.
4. Save the results.

Step 2.2: Enable the Load Balancer

To enable the load balancer, access the ESG UI and do the following:

1. Select Edge, Manage, Load Balancer, and then Global Configuration.
2. Edit load balancer global configuration.
3. Enable load balancer.
4. Enable acceleration.
5. Set logging to desired level (Info or greater).
Step 2.3: Create Application Profiles

The Application Profiles allow advanced options as well as linking to the SSL Certificate. You must create three Profiles: PCF-HTTP, PCF-HTTPS and PCF-TCP.

To create the application profiles, access the ESG UI and do the following:

1. Select Edge, Manage, Load Balancer, and then Global Application Profiles.

2. Create/Edit Profile and make the PCF-HTTP rule, turning on Insert X-Forwarded-For HTTP header.

3. Create/Edit Profile and make the PCF-HTTPS rule, same as before, but add the service certificate inserted before. If encrypting TLS traffic to the Gorouters, turn on Enable Pool Side SSL. Otherwise, leave it unchecked.
Step 2.4: Create Application Rules

In order for the ESG to perform proper x-forwarded requests, you need to add a few HAProxy directives to the ESG Application Rules. NSX supports most directives that HAProxy supports.

To create the application rules, access the ESG UI and do the following:

1. Select Edge, Manage, Load Balancer, and then Application Rules.
2. Copy and paste the table entries below into each field, one per rule.

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>option httplog</td>
<td>option httplog</td>
</tr>
<tr>
<td>reqadd X-Forwarded-Proto:\https</td>
<td>reqadd X-Forwarded-Proto:\https</td>
</tr>
</tbody>
</table>
Step 2.5: Create Monitors For Pools

NSX ships with several load balancing monitoring types pre-defined. These types are for HTTP, HTTPS and TCP. For this installation, operators build new monitors matching the needs of each pool to ensure correct 1:1 monitoring for each pool type.

To create monitors for pools, access the ESG UI and do the following:

1. Select Edge, Manage, Load Balancer, and then Service Monitoring.
2. Create a new monitor for http-routers, and keep the defaults.
3. Set the Type to HTTP.
4. Set the Method to GET.
5. Set the URL to /health.
6. Create a new monitor for tcp-routers, and keep the defaults.
7. Set the type to HTTP.
8. Set the Method to GET.
9. Set the URL to /health.
10. Create a new monitor for diego-brains, and keep the defaults.
11. Set the type to TCP.

These monitors are selected during the next step when pools are created. A pool and a monitor are matched 1:1.

Step 2.6: Create Pools of Multi-Element PCF Targets

The following steps creates the pools of resources that ESG load-balances to: the Gorouter, TCP Router, and Diego Brain jobs deployed by BOSH Director. If the IP addresses specified in the configuration do not exactly match the IP addresses reserved or used for the resources, then the pool will not effectively load balance.

Step 2.6a: Create Pool for http-routers

To create pool for http-routers, access the ESG UI and do the following:

1. Select Edge, Manage, Load Balancer, and then Pools.
2. Enter ALL the IP addresses reserved for the Gorouters into this pool. If you reserved more addresses than you have Gorouters, enter the addresses anyway and the load balancer ignores the missing resources as “down”.

   **Note:** If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address space.

3. Set the **Port** to 80 and **Monitor Port** to 8080. The assumption is that internal traffic from the ESG load balancer to the Gorouters is trusted because it is on a VXLAN secured within NSX. If using encrypted TLS traffic to the Gorouter inside the VXLAN, set the **Port** to 443.

4. Set the **Algorithm** to **ROUND-ROBIN**.

5. Set **Monitors** to **http-routers**.

---

Step 2.6b: Create Pool for **tcp-routers**

1. Select Edge, Manage, Load Balancer, and then Pools.

2. Enter ALL the IP addresses reserved for TCP Routers into this pool. If you reserved more addresses than you have VMs, enter the addresses anyway and the load balancer ignores the missing resources as “down”.

   **Note:** If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address space.

3. Set the **Port** to empty (these numbers vary) and the **Monitor Port** to 80.

4. Set the **Algorithm** to **ROUND-ROBIN**.

5. Set the **Monitors** to **tcp-routers**.

---

Step 2.6c: Create Pool for **diego-brains**

1. Select Edge, Manage, Load Balancer, and then Pools.

2. Enter ALL the IP addresses reserved for Diego Brains into this pool. If you reserved more addresses than you have VMs, enter the addresses anyway and the load balancer will just ignore the missing resources as “down”.

   **Note:** If your deployment matches the Reference Architecture for PCF on vSphere, these IP addresses are in the 192.168.20.0/22 address space.

3. Set the **Port** to 2222 and the **Monitor Port** to 2222.

4. Set the **Algorithm** to **ROUND-ROBIN**.
Step 2.7: Create Virtual Servers

This is the Virtual IP (VIP) that the load balancer uses to represent the pool of Gorouters to the outside world. This also links the Application Policy, Application Rules, and backend pools to provide PCF load balancing services. This is the interface that the load balancer balances from. You create three Virtual Servers.

1. Select Edge, Manage, Load Balancer, and then Virtual Servers.

2. Select an IP address from the available routable address space allocated to the ESG. For information about reserved IP addresses, see General Overview.

3. Create a new Virtual Server named GoRtr-HTTP and select Application Profile PCF-HTTP.
   - Use Select IP Address to select the IP address to use as a VIP on the uplink interface.
   - Set Protocol to match the Application Profile protocol (HTTP) and set Port to match the protocol (80).
   - Set Default Pool to the pool name set in Step 2.6a: Create Pool for http-routers. This connects this VIP to the pool of resources being balanced to.
   - Ignore Connection Limit and Connection Rate Limit unless these limits are desired.
   - Switch to Advanced Tab on this Virtual Server.
   - Use the green plus to add/attach three Application Rules to this Virtual Server:
     - option httplog
     - reqadd X-Forwarded-Proto:\thttp
   - **Note:** Be careful to match protocol rules to the protocol VIP-HTTP to HTTP and HTTPS to HTTPS.

4. Create a new Virtual Server named GoRtr-HTTPS and select Application Profile PCF-HTTPS.
   - Use Select IP Address to select the same IP address to use as a VIP on the uplink interface.
   - Set Protocol to match the Application Profile protocol (HTTPS) and set Port to match the protocol (443).
   - Set Default Pool to the pool name set in Step 2.6a: Create Pool for http-routers. This connects this VIP to that pool of resources being balanced to.
   - Ignore Connection Limit and Connection Rate Limit unless these limits are desired.
   - Switch to Advanced Tab on this Virtual Server.
   - Use the green plus to add/attach three Application Rules to this Virtual Server:
     - option httplog
     - reqadd X-Forwarded-Proto:\https
   - **Note:** Be careful to match protocol rules to the protocol VIP-HTTP to HTTP and HTTPS to HTTPS.

5. Create a new Virtual Server named TCPTrs and select Application Profile PCF-TCP.
   - Use Select IP Address to select the IP address to use as a VIP on the uplink interface.
   - Set Protocol to match the Application Profile protocol (TCP) and set Port to match the protocol (5000).
   - Set Default Pool to the pool name set in Step 2.6b: Create Pool for tcp-routers. This connects this VIP to the pool of resources being balanced to.
   - Ignore Connection Limit and Connection Rate Limit unless these limits are desired.
6. Create a new Virtual Server named SSH-DiegoBrains and select Application Profile PCF-HTTPS.

- Use Select IP Address to select the same IP address to use as a VIP on the uplink interface if you want to use this address for SSH access to apps. If not, select a different IP address to use as the VIP.
- Set Protocol to TCP and set Port to 2222.
- Set Default Pool to the pool name set in Step 2.6c: Create Pool for diego-brains. This connects this VIP to that pool of resources being balanced to.
- Ignore Connection Limit and Connection Rate Limit unless these limits are desired.
Step 3: Configure NAT/SNAT

The ESG obfuscates the PCF installation through network translation. The PCF installation is placed entirely on non-routable RFC-1918 network address space, so you must translate routable IP addresses to non-routable IP addresses to make connections.

<table>
<thead>
<tr>
<th>Action</th>
<th>Applied on Interface</th>
<th>Original IP</th>
<th>Original Port</th>
<th>Translated IP</th>
<th>Translated Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAT</td>
<td>uplink</td>
<td>192.168.0.0/16</td>
<td>any</td>
<td>IP_of_PCF</td>
<td>any</td>
<td>any</td>
<td>All Nets Egress</td>
</tr>
<tr>
<td>SNAT</td>
<td>infra</td>
<td>192.168.10.OpsMgr</td>
<td>any</td>
<td>IP_of_OpsMgr</td>
<td>any</td>
<td>tcp</td>
<td>OpsMgr Mask for external use</td>
</tr>
<tr>
<td>DNAT</td>
<td>infra</td>
<td>192.168.10.OpsMgr</td>
<td>any</td>
<td>192.168.10.OpsMgr</td>
<td>any</td>
<td>tcp</td>
<td>OpsMgr Mask for internal use</td>
</tr>
</tbody>
</table>

Note: The NAT/SNAT on the **infra** network in this table is an example of an optional **Hairpin NAT** rule to allow VMs within the **Infrastructure** network to access the Ops Manager API. This is because the Ops Manager hostname and the API HTTPS endpoint are registered to the Ops Manager external IP address. A pair of Hairpin NAT rules are necessary on each internal network interface that requires API access to Ops Manager. You should create these rules only if the network must access the Ops Manager API.

**Note:** Correct NAT/SNAT configuration is required for the PCF installation to function as expected.

NAT/SNAT functionality is not required if routable IP address space is used on the Tenant Side of the ESG. At that point, the ESG simply performs routing between the address segments.

**Note:** NSX generates a number of DNAT rules based on load balancing configs. You can safely ignore these.

## Additional Notes

The ESG also supports scenarios where Private RFC subnets and NAT are not utilized for **Deployment** or **Infrastructure** networks, and the guidance in
this document can be modified to meet those scenarios.

Additionally, the ESG supports up to 10 Interfaces allowing for more Uplink options if necessary.

The use of Private RFC-1918 subnets for PCF Deployment networks was chosen due to its popularity with customers. ESG devices are capable of leveraging ECMP, OSPF, BGP, and IS-IS to handle dynamic routing of customer and/or public L3 IP space. That design is out of scope for this document, but is supported by VMware NSX and Pivotal PCF.
Upgrading vSphere without PCF Downtime

Page last updated:

This topic describes how to upgrade the vSphere components that host your Pivotal Cloud Foundry (PCF) installation without service disruption.

Minimum Requirements

At a bare minimum, vSphere contains the following components:

- vCenter Server
- one or more ESXi hosts

You cannot perform an in-place upgrade of vSphere without at least two ESXi hosts in your cluster.

If you do not meet this requirement (in other words, you have insufficient resources to evacuate an entire host), then you may experience PCF downtime during the upgrade.

To upgrade vSphere with only one ESXi host or without sufficient headroom capacity, you must reduce your PCF installation size. In other words, you can either reduce the number of Diego cells in your deployment or pause PCF VMs to make more capacity available. These actions can result in PCF downtime.

Recommended Starting Configuration

If you are running a PCF deployment as recommended by the base reference architecture for PCF on vSphere (recommended), then your vSphere installation should have the following components:

- One vCenter Server
- Three ESXi hosts per cluster
- Three or more clusters
- One (or HA pair) NSX Edge appliances

Note: Pivotal recommends having at least three ESXi hosts in your cluster to maintain PCF high availability during your upgrade.

For more information, see the Reference Architecture for Pivotal Cloud Foundry on vSphere.

Procedure to Upgrade vSphere

To upgrade the vSphere management layer underneath PCF, perform the following steps:

Step 1. Upgrade vCenter

For example, you might be upgrading vCenter 6.0 to vCenter 6.5.

For more information about how to upgrade vCenter, see Overview of the vCenter Server Upgrade Process in VMware documentation.

Step 2. Upgrade ESXi Hosts

After a successful vCenter upgrade, upgrade your ESXi hosts one at a time.

Starting with the first ESXi host, perform the following steps:

1. Verify that your ESXi hosts have sufficient resources and headroom to evacuate the VM workload of a single ESXi host to the two remaining hosts.

Note: If you have enabled vSphere HA on your ESXi host, then each ESXi host should have sufficient headroom capacity since HA reserves 66% of available memory.
2. Use vMotion to move all the PCF VMs on the host you want to upgrade to the other ESXi hosts. vMotion places the VMs on the other hosts based on available capacity. For more information, see Migration with vMotion in VMware documentation.

3. Upgrade the evacuated ESXi host. For example, you may be upgrading from ESXi v6.0 to ESX v6.5. For instructions, see Upgrading ESXi Hosts in VMware documentation.

After successfully upgrading the first ESXi host, repeat the above steps for each remaining host one at a time. vSphere automatically rebalances all PCF VMs back onto the upgraded hosts via DRS after all the hosts are done.

Step 3. Upgrade ESG on VMware NSX

If your PCF deployment lives on a network behind an Edge Services Gateway (ESG) as recommended by the reference architecture, then upgrade each ESG only after completing the upgrade of vCenter and your ESXi hosts.

When you upgrade an ESG on VMware NSX, you upgrade the NSX Manager software. This upgrade can cause some slight downtime, the amount of which depends on the number of ESGs you are using.

- If your deployment only has one ESG, you can expect a downtime of 5 minutes for network reconvergence.
- If your ESGs are deployed in HA, upgrade the first ESG. Then upgrade the second ESG. This upgrade results only in 15-20 seconds of downtime.

For more information, see the NSX Upgrade Guide in VMware documentation.
Migrating PCF to a New Datastore in vSphere

This topic describes how to migrate your Pivotal Cloud Foundry (PCF) installation to a new vSphere datastore.

Prerequisites

Both the new and existing vSphere datastores must reside in the same datacenter.

To avoid service disruption, Pivotal recommends that you configure your overall PCF deployment for high availability. In addition, check for configurations necessary to achieve high availability in each of your installed product tiles.

If your environment has any single points of failure, service may be disrupted as a result of the migration.

Before You Begin

This section describes the steps you should perform prior to the migration.

Step 1: Back Up Your Environment

Ensure that your PCF environment is fully backed up.

For more information about how to backup PCF, see Backing up Pivotal Cloud Foundry.

Step 2: Document Current Environment Settings

Document your current environment settings before proceeding with the datastore migration. Record which VMs are running and in which datastore they reside. If you experience any issues during or after the migration, you must have this information to restore your environment.

To obtain this information, perform the following steps:

1. Run the `bosh instances` command.

   - If you use BOSH CLI v2+, run the following command, replacing `MY-ENV` with the alias you assigned to your BOSH Director:

     ```bash
     $ bosh -e MY-ENV instances --details > instances.txt
     ```

   - If you use the original version of the BOSH CLI, run the following command:

     ```bash
     $ bosh-old instances --details > instances.txt
     ```

2. Save the resulting file `instances.txt` to a safe location.

3. Note the datastore where each VM resides in vSphere.

Step 3: Check System Health

In Pivotal Application Service (PAS), check the Status tab and make sure there are no errors or reported issues.

Step 4: Check Installed Products Health

In each tile installed in your PCF deployment, check the Status tab and make sure there are no errors or reported issues.
Step 5: Check BOSH Director Status

1. Check that there are no pending changes and that the status of all tiles is green.
2. Make sure the last Installation Log does not contain any errors.
3. Before proceeding with the migration, click Review Pending Changes and Apply Changes to make sure there are no errors in the Installation Log.

Procedure: Migrate PCF to a New Datastore

1. In BOSH Director, navigate to the vCenter Config page.
2. Update the Ephemeral Datastore Names and Persistent Datastore Names field to reflect the new datastore names, then click Save.
   
   Note: If you use the Datastore Clustering feature in vSphere, provide only the individual names of the datastores in the cluster. Do not provide the name of the cluster that contains them.

3. Click Review Pending Changes, then Apply Changes.
4. Confirm that the BOSH Director VM has persistent disk on the new datastore.
   a. Navigate to vCenter Resource Pools and select the Resource Pool that contains your PCF deployment VMs and new datastore.
   b. Click the Related Objects and Virtual Machines.
   c. Locate the Ops Manager VM and verify that the VM has an expected value in the Provisioned Space column.

5. In BOSH Director, navigate to the Director Config page, and select the Recreate all VMs option.
6. Click Review Pending Changes, then Apply Changes.

After the Migration

When BOSH moves disks, it waits for up to 60 minutes for the operation to complete. If the operation does not complete in time, BOSH can enter a state where it claims that the disks are out of sync.

Fix Failed BOSH Deployment with Out-of-Sync Error

If your PCF deployment gets into this state, you can resolve the issue by performing the steps in the KB article How to recover from a failed bosh deployment when VMs are out of sync on vSphere.

Prevent Out-of-Sync Error

You can also prevent the out of sync BOSH error by increasing the CPI timeout to a larger value before performing the migration. Follow the instructions in the KB article How to Increase the Timeout on Bosh CPI Command Calls.
Control Plane Reference Architectures

Introduction

Concourse is the main continuous integration and continuous delivery (CI/CD) tool that the Pivotal and open-source Cloud Foundry communities use to develop, test, deploy and manage Cloud Foundry foundations.

This topic describes topologies and best practices for deploying Concourse on BOSH, and using Concourse to manage Pivotal Cloud Foundry (PCF) foundations. For production environments, Pivotal recommends deploying Concourse with BOSH.

The three topologies described in this document govern the network placement and relationship between two systems:

- The control plane runs Concourse to gather sources for, integrate, update, and otherwise manage PCF foundations. This layer may also host internal Docker registries, S3 buckets, git repositories, and other tools.
- Each PCF foundation runs PCF on an instance of BOSH.

Each topology described below has been developed and validated in multiple PCF customer and Pivotal Labs environments.

Deployment Topologies

Security policies are usually the main factor that determines which CI/CD deployment topology best suits a site's needs. Specifically, the decision depends on what network connections are allowed between the control plane and the PCF foundations that it manages.

The following three topologies answer a range of security needs, ordered by increasing level of security around the PCF foundations:

- **Topology 1**: Concourse server and worker VMs all collocated on control plane
- **Topology 2**: Concourse server on control plane, and remote workers collocated with PCF foundations
- **Topology 3**: Multiple Concourse servers and workers collocated with PCF foundations

Across these three topologies, the increasing level of PCF foundation security correlates with:

- Increasing network complexity
- Increasing effort required for initial deployment and ongoing maintenance

Security Decision Factors

The graph below captures the decision factors dictated by network security policy, and recommends Concourse deployment topologies that adhere to those policies.

![Decision Factors Diagram]

**Additional Decision Factors**

In addition to security policy, deciding on a CI/CD deployment topology may also depend on factors such as:

- Network latency across network zones
Air-gapped vs. Internet-connected environments

Resource limitations

The Credentials Management and Storage Services sections below includes notes and recommendations regarding credentials management, Docker registries, S3 buckets, and git repositories, but does not cover these tools extensively.

Topology Objects

Complete deployment topologies dictate the internal placement or remote use of the following:

- Concourse server and worker VMs, as discussed above
- Credentials managers such as CredHub or Vault
- Docker registries, public or private
- S3 or other storage buckets, public or private
- git or other code repositories, public or private

Topology 1: Concourse Server and Worker VMs All Colocated on Control Plane

This simple topology follows network security policies that allow a single control plane to connect to all PCF foundations deployed across multiple network zones or data centers.

Connectivity Requirements

Concourse worker VMs must be allowed to connect to:

- The Ops Manager VM or a jumpbox VM in all of the PCF foundations networks
- (on vSphere) The vCenter API for each PCF foundation

Performance Notes

Network data transfers between the control plane and each PCF foundation network zone may carry large files such as PCF tiles, release files, and PCF foundation backup pipelines output.

Pivotal recommends that you test network throughput and latency between those network zones to make sure that data transfer rates do not make pipeline execution times unacceptably long.

Firewall Requirements

All Concourse worker VMs are required to connect to certain VMs on PCF foundation subnets across networks zones or data centers.

See PCF CI/CD Pipelines section for required VMs, ports, and external websites required for PCF CI/CD pipelines.

Pros
• Simplified deployment and maintenance of centralized control plane, which requires only one BOSH deployment and runs one BOSH Director.
• Simplified setup and maintenance of PCF CI/CD pipelines. All pipelines and Concourse teams use a single, centralized server.

Cons
• You may have to configure firewall rules in each PCF network to allow connectivity from workers in CI/CD zone, as mentioned above.

Topology 2: Concourse Server on Control Plane, and Remote Workers Colocated with PCF Foundations

This topology supports environments where PCF foundation VMs cannot receive incoming traffic from IP addresses outside their network zone or data center, but they can initiate outbound connections to outside zones.

As in Topology 1, the Concourse server, and potentially other VMs for tools such as Docker registries or S3 buckets, are deployed to a dedicated subnet. The difference here is that Concourse worker VM pools are deployed inside each PCF foundation subnet, network zone, or data center.

Connectivity Requirements
Concourse worker VMs in each PCF foundation network zone or data center must:
• Connect to the Ops Manager VM or a jumpbox VM in each PCF foundations network
• (on vSphere) Connect to the vCenter API for each PCF foundation
• Have outbound connection access to the Concourse web/ATC server on port 2222. This lets them handshake with the Concourse server to open a reverse SSH tunnel for ongoing communication between them and the Concourse ATC.

For more details, see Concourse Architecture.

Performance Notes
Remote workers have to download large installation files from either the internet or from a configured S3 artifacts repository. For PCF backup pipelines, workers may also have to upload large backup files to the S3 repository.

Pivotal recommends that you test network throughput and latency between those network zones to make sure that data transfer rates do not make pipeline execution times unacceptably long.

Firewall Requirements
Remote worker VMs require outbound access to the Concourse web/ATC server on port 2222.

Remote worker VMs inside each PCF foundation network zone or data center are required to connect to VMs in their colocated foundation. They may also require access to external web sites or S3 repositories for downloading installation files.

See the PCF CI/CD Pipelines section for required VMs and ports required for PCF CI/CD pipelines.

Pros
Relatively simple maintenance of centralized control plane, which contains single Concourse server and other tools.

Simplified setup and maintenance of PCF CI/CD pipelines. All pipelines and Concourse teams use a single, centralized server.

Cons

- You have to reconfigure firewalls to grant outbound access to remote worker VMs.
- In addition to deploying the control plane, you need an additional BOSH deployment and running BOSH Director for each PCF Foundation network zone or data center.
- You need to manage multiple Concourse worker pools in multiple locations.

Topology 3: Multiple Concourse Servers and Workers Colocated with PCF Foundations

This topology supports environments where PCF foundation VMs can only be accessed from within the same network zone or data center. This scenario requires deploying complete and dedicated control planes within each deployment zone.

Performance Notes

Since workers run in the same network zone or data center as the PCF foundation, data transfer throughput should not limit pipeline performance.

Firewall Requirements

- **Air-gapped environments** require some way to bootstrap S3 repository with Docker images and PCF releases files for pipelines from external sites.
- **Non-air-gapped environments**, in which workers can download required files from external websites, need those websites to be whitelisted in the proxy or firewall setup.

Pros

- Requires little or no firewall rules configuration for control plane VMs. In non-air-gapped environments, worker VMs download PCF releases and Docker images for pipelines from external websites.

Cons

- Requires deploying and maintaining multiple Concourse and other tools.
- Requires deploying and maintaining multiple PCF pipelines for each Concourse server.
- For air-gapped environments, requires setting up an S3 repository for each control plane.

Deploying CI/CD to the Control Plane

There are a few alternatives to deploy BOSH Directors, Concourse servers, and other tool releases to the control plane. Details on those alternatives are outside of the scope of this document, but refer to the links below for the most common options:

Manual deployments

- **BOSH Director**: BOSH create-env
- **Concourse**
  - Concourse
  - Concourse with CredHub
  - Concourse with Vault
Docker registries

- Private Docker Registry
- VMware Harbor Registry

S3 buckets

- Minio S3
- EMC Cloud Storage

Automated deployments

- Bosh Bootloader (BBL) deploys BOSH Director and Concourse on multiple IaaSes.

Control Plane Deployment Best Practices

Here are some best practices for control plane components that apply across all deployment topologies:

Dedicated BOSH Director

Pivotal recommends deploying Concourse and other control plane tools on their own, dedicated BOSH layer, with their own BOSH Director that runs separately from the PCF foundations that the control plane manages.

Pivotal does not recommend using an existing PCF BOSH Director instance to deploy Concourse and other software (e.g. Minio S3, private Docker registry, CredHub). Sharing the same BOSH Director with PCF deployments increases the risk of accidental or undesired updates or deletion of those deployments.

Dedicating a BOSH Director to Concourse other control plane tools also provides higher flexibility for PCF foundation upgrades and updates, such as stemcell patches.

Credentials Management

All credentials and other sensitive information that feeds into a Concourse pipeline should be encrypted and stored using credentials management software such as CredHub or Vault. Never store credentials as plain text in parameter files in file repositories.

Credentials Management with CredHub

Concourse integrates with CredHub to manage credentials in its pipelines. The pipelines reference encrypted secrets stored in a CredHub server and retrieve them automatically during execution of tasks.
To integrate Concourse with a CredHub server, you configure its ATC job’s deployment properties with information about the CredHub server and corresponding UAA authentication credentials.

You can deploy CredHub in multiple ways: as a dedicated VM, or integrated with other VMs, such as colocating the CredHub server with the BOSH Director VM or Concourse’s ATC/web VM.

Colocating a CredHub server with Concourse’s ATC VM dedicates it to the Concourse pipelines and lets Concourse administrators manage the credentials server. This configuration also means that during Concourse upgrades, the CredHub server only goes down when the Concourse ATC job is also down, which minimizes potential credential server outages for running pipelines.

The diagram below illustrates the jobs of Concourse VMs, along with the ones for the BOSH Director VM, when a dedicated CredHub server is deployed with Concourse.

The [Concourse Pipelines Integration with CredHub](#) documentation in the PCF Pipelines repository describes how to deploy a CredHub server integrated with Concourse.

Credentials Management with Vault

For how to configure Vault to manage credentials for Concourse pipelines, see [Secure credential automation with Vault and Concourse](#) in the PCF Pipelines repository.

Storage Services

Git Server

BOSH and Concourse implement the concepts of infrastructure-as-code and pipelines-as-code. As such, it is important to store all source code for deployments and pipelines in a version-controlled repository.

PCF CI/CD pipelines assume that their source code is kept in git-compatible repositories. [GitHub](#) is the most popular git-compatible repository for Internet-connected environments.
GitLab, BitBucket and GOGS are examples of git servers that can be used for both connected and air-gapped environments.

A git server that contains configuration and pipeline code for a PCF foundation needs to be accessible by the corresponding worker VMs that run CI/CD pipelines for that foundation.

S3 Repository

In all environments Concourse requires an S3 repository to store PCF backups and possibly other files. If your deployment requires a private, internal S3 repository but your IaaS lacks built-in options, you can use BOSH to deploy your own S3 releases, such as Minio S3 and EMC Cloud Storage to your control plane.

For air-gapped environments, an S3 repository is also the preferred method to store release files for PCF tiles, stemcells and buildpacks. Docker images can also be stored to an S3 repository as an alternative to a private Docker registry. See Offline Pipelines for Airgapped Environments for details.

Private Docker Registry

For air-gapped environments, Docker Images for Concourse pipelines need to be stored either on a private Docker registry or in an S3 repository. For BOSH-deployed private registry alternatives, check Docker Registry or VMWare Harbor.

High Availability

For details on how to set up a load balancer to handle traffic across multiple instances of the ATC/web VM, and how to deploy multiple worker instances, see the Concourse Architecture topic.

PCF CI/CD Pipelines

PCF Platform Automation with Concourse (PCF Pipelines) is a collection of Concourse pipelines for installing and upgrading Pivotal Cloud Foundry. See the source on Github or download from Pivotal Network (sign-in required).

⚠️ warning: At time of publication, the PCF Pipelines repository is undergoing planned deprecation.

To run PCF Pipelines, you need:

- Ops Manager web UI, API and VM installed
- (vSphere environments) vCenter API installed
- For proxy- or Internet-connected environments, whitelist the following sites:
  - Docker Hub: hub.docker.io
  - Pivotal Network: network.pivotal.io
  - bosh.io

BOSH Backup and Restore (BBR) PCF Pipelines

BBR PCF Pipelines automate PCF foundation backups. See and download the source on Github.

To run BBR PCF Pipelines, you need an S3 repository for storing backup artifacts.

Note: BBR PCF Pipelines is a PCF community project not officially supported by Pivotal.

Pipelines Orchestration Frameworks

PCF Pipelines Maestro uses a Maestro framework to:

- Automate pipeline creation and management for multiple PCF foundations
- Promote and audit configuration changes and version upgrades across all foundations

Note: PCF Pipelines Maestro is a PCF community project not officially supported by Pivotal.
Concourse Team Management Best Practices

When a single Concourse server hosts CI/CD pipelines for more than one PCF foundation, Pivotal recommends creating one Concourse team (not main) specific to each foundation, and associating that team with all pipelines for that foundation, e.g. install, upgrade, backup, and metering.

Dedicating a Concourse team to each PCF foundation has the following benefits:

- **It avoids the clutter of pipelines in a single team.** The list of pipelines for each foundation may be long, depending how many tiles are deployed to it.
- **It avoids the risk of operators running a pipeline for the wrong foundation.** When a single team hosts maintenance pipelines for multiple foundations, the clutter of dozens of pipelines may lead operators to accidentally run a pipeline (e.g. upgrade or delete tile) targeted at the wrong PCF foundation.
- **It allows for more granular access control settings per team.** PCF pipelines for higher environments (e.g. Production) may require a more restricted access control than ones from lower environments (e.g. Sandbox). Authentication settings for Concourse teams enable that level of control.
- **It allows workers to be assigned to pipelines of a specific foundation.** Concourse deployment configuration allows for the assignment of workers to a single team. If that team contains pipelines of only one foundation, then the corresponding group of workers run pipelines only for that foundation. This is useful when security policy requires tooling and automation for a foundation (e.g. Production) to run on specific VMs.

Concourse Teams for App Development Orgs and Spaces

When a Concourse server hosts pipelines for an app development team, Pivotal recommends creating a Concourse team associated with that development team, and associating Concourse team membership with org and space membership defined in the Pivotal Application Service (PAS) user authentication and authorization (UAA) server.

Associating Concourse teams with PAS org and space member lists synchronizes access between PAS and Concourse, letting developers see and operate the build pipelines for the apps they develop.
Global DNS Load Balancers for Multi-Foundation Environments

This topic describes global DNS load balancers (GLBs) for multi-foundation environments. This topic also describes concepts such as foundation affinity and healthchecks.

Note: If you want to configure a load balancer dedicated to one PCF foundation and you are using an F5 LTM, see Configuring an F5 Load Balancer for PAS.

About Multi-Foundation Environments

Multi-foundation environments are multiple instances of BOSH and Ops Manager that can communicate with each other. Each foundation can use different infrastructures to fit your preferences.

Multi-foundation environments are commonly deployed in an active-active or active-passive pattern. See below for descriptions of each term:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active-active pattern</td>
<td>Instances of apps run on two foundations and they are both in use.</td>
</tr>
<tr>
<td>Active-passive pattern</td>
<td>Instances of apps run on two foundations, but the instances may only be active on one foundation. The other foundation becomes active only in the event of a failover.</td>
</tr>
</tbody>
</table>

Configure Your GLB

The typical setup for foundation failover requires the GLB be authoritative for a wildcard app domain. The wildcard app domain is not the same domain as the foundation-default app domain.

To configure your GLB, do the following:

- Find and record your wildcard app domain. For Pivotal Application Service (PAS), your wildcard app domain is typically the Apps Domain you configure in the Domains pane of the PAS tile.
- Add the domain you recorded to both PCF foundations using a shared domain. You can create a shared domain using the Cloud Foundry Command Line Interface (cf CLI). For more information about the cf CLI command to create shared domains, see create-shared-domain in the cf CLI reference guide.
- To support failover, set the time-to-live (TTL) in the wildcard DNS record. Set the TTL to about 30 to 180 seconds. When determining your TTL, consider the tradeoff between the app performance impact and the resulting time for failover to occur.

About Foundation Affinity

Foundation affinity occurs when the GLB favors one foundation over another during a route request. For example, users experience less latency if they are routed to a foundation that is geographically closer, so the GLB may favor that foundation.

Different GLBs have their own mechanisms to achieve this. See the following table for common concepts:

<table>
<thead>
<tr>
<th>Foundation Affinity Concepts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>Topology/geographically-based affinity</td>
<td>The GLB attempts to direct traffic to the graphically nearest foundation based on IP geolocation or provided topology for private networks of the LDNS server performing the lookup.</td>
</tr>
<tr>
<td>Static-persist/member-hashing affinity</td>
<td>The GLB attempts to direct traffic to the graphically nearest foundation based on IP geolocation or provided topology for private networks of the LDNS server performing the lookup.</td>
</tr>
<tr>
<td>Active/passive foundations</td>
<td>If one foundation is usually idle, you can always pick the active foundation IP as long as it remains available.</td>
</tr>
</tbody>
</table>
| Microservice-to-microservice affinity | For microservice apps, you typically use a Services Registry to manage traffic between microservices. There are two ways to do this:  
  - If the microservices are using the domain which maps to the GLB, then their traffic is routed through the GLB.  
  - If the microservices are communicating using IP address or internal domains, such as when using Envoy, then the... |
About Healthchecks

Healthchecks determine whether a foundation for an app is healthy or not.

See the following table for the levels at which you can check the health of your foundation:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foundation</td>
<td>GLB relies on a local load balancer in front of the PCF Gorouters to determine the overall health of the foundation. GLB can perform healthchecks on TCP port 443 or 80 or on the local load balancer. The local load balancer healthchecks the backend pool of Gorouters on port 8080.</td>
</tr>
<tr>
<td>App</td>
<td>Healthchecks are set only for apps which have instances on both foundations. Each app instance has canary DNS records. The canary DNS records are the same for the app instances on each foundation. You would also need to add more VIPs dedicated to these canary apps that would be used to check their health.</td>
</tr>
</tbody>
</table>

⚠️ warning Pivotal does not recommend setting healthchecks at the app level. Configuring healthchecks in this way can cause more frequent failover and delays while pushing apps. Complete failover of a foundation affects all apps on the platform. Healthchecks on a per-app basis can require additional overhead beyond the control of an app developer.
Backing Up and Restoring Pivotal Cloud Foundry

Page last updated:

Consider the following when backing up data in your Pivotal Cloud Foundry (PCF) deployment:

- If your deployment uses external databases, check that it is a supported database configuration.
- If your PCF deployment uses internal databases, follow the backup and restore instructions for the MySQL server included in the BBR documentation.

General Data Protection Regulation

The General Data Protection Regulation (GDPR) came into effect on May 25, 2018 and impacts any company processing the data of EU citizens or residents, even if the company is not EU-based. The GDPR sets forth how companies should handle privacy issues, securely store data, and respond to security breaches.

Backup artifacts may contain personal data covered by GDPR. For example, a backup of a PAS could contain a user email. For further information regarding personal data that may be stored in PCF, see here.

Backup and Restore with BBR

BOSH Backup and Restore (BBR) is a command-line tool for backing up and restoring BOSH deployments.

To perform a backup of your PCF deployment with BBR, see Backing Up Pivotal Cloud Foundry with BBR.

To restore your PCF deployment with BBR, see Restoring Pivotal Cloud Foundry from Backup with BBR.

To troubleshoot problems with BBR, see Troubleshooting BBR.

BBR Restore Scenarios

There are many different restore cases that can arise with BOSH Directors and PCF deployments. We have provided some guidance on how to use BBR in common restore scenarios below:

- Restoring Lost BOSH Director Persistent Disk
- Restoring Lost PAS Data
- Recovering from an Ops Manager and PAS Upgrade Failure

These guides do not cover all possible restore scenarios. If you encounter another restore case, then some information in these guides may be applicable. Contact Pivotal Support for guidance.
Incident Response Guides

This guide describes how Pivotal Cloud Foundry (PCF) operators can use BOSH Backup and Restore (BBR) to respond to upgrade failures and data loss within their deployment.

Overview

This guide includes the following topics:

- Restoring Lost BOSH Director Persistent Disk
- Restoring Lost PAS Data
- Recovering from an Ops Manager and PAS Upgrade Failure
Restoring Lost BOSH Director Persistent Disk

This guide explains how to use BOSH Backup and Restore (BBR) to restore a BOSH Director that has lost its persistent disk.

Prerequisite

Restoring a BOSH Director persistent disk with BBR is only possible if you have previously taken a backup using the following command:

```
bb director backup
```

This command produces a backup artifact in the format `DIRECTOR_IP_TIMESTAMP/`.

⚠️ warning: BOSH Director backups require identical references to all BOSH-deployed VMs. If BOSH recreates your deployment VMs for any reason, such as changes to stemcells, networks, or availability zones (AZs), your BBR backup artifact is no longer compatible for performing a restore.

Prepare Your Environment

Your BOSH Director must be in a healthy state before you can perform a BOSH Director restore. The BOSH Director is in a healthy state when all jobs are running.

If your BOSH Director has lost its persistent disk, create a new disk to return your BOSH Director to a healthy state.

Create a New BOSH Director Persistent Disk

To create a new persistent disk for the BOSH Director, do the following:

1. Create a new persistent disk using your IaaS console.
2. Log in to the Ops Manager VM. See the [Log in to the Ops Manager VM with SSH](#) section for your IaaS in Advanced Troubleshooting with the BOSH CLI.
3. Open the `bosh-state.json` file in a text editor. Edit the `disks.cid` value to match the new persistent disk you created in the first step.
4. Update the persistent disk size in Ops Manager and then click Apply Changes. This creates a new BOSH Director VM and attaches the new persistent disk to it.

Perform the Restore

⚠️ warning: BBR restore is a destructive process which removes any current data in your deployment. Performing a restore overwrites all new data since you created your most recent backup artifact.

Perform a BOSH Director restore by following the BBR instructions in [Step 8: Restore the BOSH Director](#) of Restoring PCF From Backup with BBR.

Consolidate Your Environment

After you successfully restore your BOSH Director, the BOSH Director uses the VM references that were stored in the backup artifact. If you clicked Apply Changes in Ops Manager between when the backup and the restore took place, check if any BOSH-deployed VMs were deleted or added.

If changes have occurred, ensure that the BOSH Director database is consistent with the current state of your IaaS after the restore. If VMs were deleted, the BOSH Director still has references to these deleted VMs after the restore. If VMs were added, the BOSH Director has no knowledge of the new VMs.

To ensure that the BOSH Director database matches your IaaS console, do the following:

1. Run the following command to find and delete outdated VM references for an environment:
bosh --deployment DEPLOYMENT-NAME cloud-check

Where `DEPLOYMENT-NAME` is the name of your BOSH deployment.

2. Log in to your IaaS console and delete any orphaned VMs that your BOSH Director does not reference.

Your BOSH Director should now be in a healthy state with the new persistent disk attached. Your BOSH deployment VMs should be aligned with the current state of your IaaS.
Restoring Lost PAS Data

This guide explains how to use BOSH Backup and Restore (BBR) to restore a piece of Pivotal Application Service (PAS) data, such as a user or space, which has been lost.

Prerequisite

Restoring PAS data with BBR is only possible if you have previously taken a backup using the following command:

```
bbr director backup
```

This command produces a backup artifact in the format `DIRECTOR_IP_TIMESTAMP/`.

Validate Your Backup Artifact

Before restoring, verify that the backup artifact you have is suitable for your restore need. You can `untar` and explore the PAS backup artifact to verify that it contains the data that you need to restore.

Use any the checks in this section depending on the data you need to restore.

Verify an Org Is Present

To verify that your backup artifact contains the org you need to restore, do the following:

```
tar -xvf backup_restore-0-bbr-cloudcontrollerdb.tar
cat cloudcontroller-artifact-file | grep "ORG-TO-RESTORE"
```

Where `ORG-TO-RESTORE` is the org you need to restore.

If you see the following line in the output of the above command, then you can restore this org:

```
INSERT INTO `organizations` VALUES (...,'ORG-TO-RESTORE',...);
```

Verify a Space Is Present

To verify that your backup artifact contains the space you need to restore, do the following:

```
tar -xvf backup_restore-0-bbr-cloudcontrollerdb.tar
cat cloudcontroller-artifact-file | grep "SPACE-TO-RESTORE"
```

Where `SPACE-TO-RESTORE` is the space you need to restore.

If you see the following line in the output of the above command, then you can restore this space:

```
INSERT INTO `spaces` VALUES (...,'SPACE-TO-RESTORE',...);
```

Verify a User Is Present

To verify that your backup artifact contains the user you need to restore, do the following:

```
tar -xvf backup_restore-0-bbr-uaadb.tar
cat uaadb-artifact-file | grep "USER-TO-RESTORE"
```

© Copyright Pivotal Software Inc, 2013-2019
Where **USER-TO-RESTORE** is the user you need to restore.

If you see the following line in the output of the above command, then you can restore this user:

```sql
INSERT INTO 'users' VALUES (...,'USER-TO-RESTORE',...);
```

**Verify a Route Is Present**

To verify that your backup artifact contains the route you need to restore, do the following:

```bash
tar -xvf backup_restore-0-bbr-cloudcontrollerdb.tar
cat cloudcontroller-artifact-file | grep "ROUTE-TO-RESTORE"
```

Where **ROUTE-TO-RESTORE** is the route you need to restore.

If you see the following line in the output of the above command, then you can restore this user:

```sql
INSERT INTO 'routes' VALUES (...,'ROUTE-TO-RESTORE',...);
```

**Verify a Domain Is Present**

To verify that your backup artifact contains the domain you need to restore, do the following:

```bash
tar -xvf backup_restore-0-bbr-cloudcontrollerdb.tar
cat cloudcontroller-artifact-file | grep "DOMAIN-TO-RESTORE"
```

Where **DOMAIN-TO-RESTORE** is the domain you need to restore.

If you see the following line in the output of the above command, then you can restore this domain:

```sql
INSERT INTO 'domains' VALUES (...,'DOMAIN-TO-RESTORE',...);
```

**Verify an App Is Present**

To verify that your backup artifact contains the app you need to restore, do the following:

```bash
tar -xvf backup_restore-0-bbr-cloudcontrollerdb.tar
cat cloudcontroller-artifact-file | grep "APP-TO-RESTORE"
```

Where **APP-TO-RESTORE** is the app you need to restore.

To verify that your blobstore backup artifact contains the droplet for the app you need to restore, do the following:

```bash
tar -xvf blobstore-backup.tar
tree blobstore-backup | grep "APP-DROPLET"
```

Where **APP-DROPLET** is the droplet for the app you need to restore.

If you see the following line in the output of the above command, then you can restore this app:

```sql
INSERT INTO 'apps' VALUES (...,'APP-TO-RESTORE','APP-DROPLET',...);
```

**Note:** If you use an external blobstore, go to your IaaS console and check the backup bucket to verify that the droplet is present.
### Perform the Restore

**warning:** BBR restore is a destructive process which removes any current data in your deployment. Performing a restore overwrites all new data since you created your most recent backup artifact.

Pivotal recommends that you verify the compatibility of your backup artifact by attempting the restore on an environment that is identical to the target environment. If this is not possible, refer to the Compatibility of Restore section of Restoring PCF from Backup with BBR to confirm that your backup is compatible with your restore environment.

Run a PAS restore by following the BBR instructions in Step 13: Restore PAS of Restoring PCF from Backup with BBR.

After a restore, PAS should roll back to its state at the time when the backup was taken, and contain the data that you had previously lost.

### Consolidate Your Environment

If you have service instances that were not present when you performed the backup, then the BOSH service deployments become orphaned. PAS has no reference to these service deployments, but they continue to exist. You can locate orphaned service deployments using the BOSH command-line interface (CLI).

You can locate orphaned on-demand service instances by running the `orphan-deployments` errand. For more information, see Orphan Deployments in Broker and Service Management.
Recovering from an Ops Manager and PAS Upgrade Failure

This topic explains how to use BOSH Backup and Restore (BBR) to restore Ops Manager and Pivotal Application Service (PAS) back to a healthy state after a failed upgrade between PAS minor versions.

Note: BBR can only restore across the same PAS minor versions. See metadata/cf-VERSION.yml in the backup artifact to verify which version of PAS was used to create the backup artifact.

Evaluate Your Deployment

You can restore Ops Manager and PAS to a healthy state after a failed upgrade between PAS minor versions if your backups, BOSH Director, and PAS deployment meet certain conditions. Follow the procedures in this section to confirm that your deployment can be restored by BBR.

Review Your Backup

Before you perform a restore, you must do the following:

1. Run the following `bbr` commands from your Ops Manager VM:
   - Back up your deployments by running the following command:
     ```
     bbr deployment --deployment DEPLOYMENT_NAME backup
     ```
   - Back up your BOSH Director by running the following command:
     ```
     bbr director backup
     ```

   The above `bbr` commands produce the following backup artifacts:
   - DEPLOYMENT_NAME_TIMESTAMP/
   - DIRECTOR_IP_TIMESTAMP/
   - installation.zip

2. Export your Ops Manager settings. For more information, see the Settings Page section of Using the Ops Manager Interface.

Validate Your Environment for Restore

Before you can restore after a failed upgrade between PAS minor versions, you must confirm that your BOSH Director and PAS deployment are healthy.

Confirm That Your BOSH Director Is Healthy

To check that the BOSH Director is in a healthy state for a `bbr restore`, SSH to the Ops Manager VM and ensure that all jobs are running.

To make sure that all jobs are running, do the following:

1. SSH into your BOSH Director. For more information, see Log in to the BOSH Director VM with SSH in Advanced Troubleshooting with the BOSH CLI.

2. View all deployment job instances by running the following command:
   ```
   monit summary
   ```

   Monit should report that all jobs are running.

3. Ensure that the BOSH API is responsive by running the following command:
If this command runs successfully, then the BOSH Director API is in a healthy and responsive state and can be restored by BBR.

If your BOSH Director is unhealthy, see Restoring an Unhealthy BOSH Director below.

Confirm That Your PAS Deployment Is Healthy

To confirm that PAS is in a healthy state, run the following command:

```
bosh -d PAS-DEPLOYMENT-NAME instances
```

Where `PAS-DEPLOYMENT-NAME` is the name of your PAS BOSH deployment.

If your PAS deployment is healthy, then the output states that all of your instances are in the `running` state. If all instances are running, PAS can be restored by BBR.

If your PAS deployment is unhealthy, see Restoring an Unhealthy PAS Deployment.

Restore Your Deployment

After you confirm that your BOSH Director is healthy with its data intact, and all instances in your PAS deployment are responsive, you can restore PAS with BBR.

Restoring PAS

If your BOSH Director is healthy, you can restore across the same PAS minor version by following the procedures in Step 13: Restore PAS in Restoring PCF from Backup with BBR.

Recover to a Healthy State

If either the BOSH Director or PAS are not in a healthy state, they must be recovered before you can run `bbr restore`. If you need guidance on the following subsections, contact Pivotal Support.

Restoring an Unhealthy BOSH Director

If the BOSH Director is not in a healthy state, Pivotal recommends following our full restore workflow to regain your Ops Manager, BOSH Director, and BOSH deployments.

See Restore Workflow in Restoring PCF from Backup with BBR.

Restoring an Unhealthy PAS Deployment

If your BOSH Director is healthy, you can restore your previous BOSH Director state by following Step 8: Restore the BOSH Director in Restoring PCF from Backup with BBR.

After your BOSH Director is restored, you can then restore your deployments by following Step 13: Restore PAS in Restoring PCF from Backup with BBR.
Disaster Recovery in Pivotal Cloud Foundry

This document provides an overview of the options and considerations for disaster recovery in Pivotal Cloud Foundry (PCF).

Operators have a range of approaches for ensuring they can recover Pivotal Cloud Foundry, apps, and data in case of a disaster. The approaches fall into the following two categories:

- Using data from a backup to restore the data in the PCF Deployment. See Back up and Restore Using BOSH Backup and Restore (BBR) for more information.
- Recreating the data in PCF by automating the creation of state in PCF. See Disaster Recovery by Recreating the Deployment for more information.

Back up and Restore using BOSH Backup and Restore (BBR)

What is BBR?

BOSH Backup and Restore (BBR) is a CLI for orchestrating backing up and restoring BOSH deployments and BOSH Directors. BBR triggers the backup or restore process on the deployment or Director, and transfers the backup artifact to and from the deployment or Director.

Use BOSH Backup and Restore to reliably create backups of core PCF components and their data. These core components include CredHub, UAA, BOSH Director, and PAS.

Each component includes its own backup scripts. This decentralized structure helps keep scripts synchronized with the components. At the same time, locking features ensure data integrity and consistent, distributed backups across your deployment.

For more information about the BBR framework, see BOSH Backup and Restore in the open source Cloud Foundry documentation.

Backing up PCF

Backing up PCF requires backing up the following components:

- Ops Manager settings
- BOSH Director, including CredHub and UAA
- Pivotal Application Service (PAS)
- Data services

For more information, see Backing up Pivotal Cloud Foundry with BBR. With these backup artifacts, operators can recreate PCF exactly as it was when the backup was taken.

Restoring PCF

The restore process involves creating a new PCF deployment starting with the Ops Manager VM. For more information, see Restoring Pivotal Cloud Foundry from Backup with BBR.

The time required to restore the data is proportionate to the size of the data because the restore process includes copying data. For example, restoring a 1 TB blobstore takes one thousand times as long as restoring a 1 GB blobstore.

Benefits

Unlike other backup solutions, using BBR to back up PCF enables the following:

- **Completeness**: BBR supports backing up BOSH, including releases, CredHub, UAA, and service instances created with an on-demand service broker. With PCF v1.12, Ops Manager export no longer includes releases.
- **Consistency**: BBR provides referential integrity between the database and the blobstore because a lock is held while both the database and blobstore are backed up.
- **Correctness**: Using the BBR restore flow addresses C2C and routing issues that can occur during restore.
API Downtime During Backups

Apps are not affected during backups, but certain APIs are unavailable. The downtime occurs only while the backup is being taken, not while the backup is being copied to the jumpbox.

In a consistent backup, the blobs in the blobstore match the blobs in the Cloud Controller Database. To take a consistent backup, changes to the data are prevented during the backup. This means that the CF API, Routing API, Usage Service, Autoscaler, Notification Service, Network Policy Server, and CredHub are unavailable while the backup is being taken. UAA is in read-only mode during the backup.

Backup Timings

The first three phases of the backup are lock, backup, and unlock. During this time, the API is unavailable. The drain and checksum phase starts once the backup scripts finish. BBR downloads the backup artifacts from the instances to the BBR VM, and performs a checksum to ensure the artifacts are not corrupt. The size of the blobstore significantly influences backup time.

The table below gives an indication of the downtime that you can expect. Actual downtime varies based on hardware and PCF configuration. These example timings were recorded with Pivotal Application Service (PAS) deployed on Google Cloud Platform (GCP) with all components scaled to one and only one app pushed.

<table>
<thead>
<tr>
<th>API State</th>
<th>Backup phase</th>
<th>Duration for External Versioned S3-Compatible Blobstore</th>
<th>Duration for External Unversioned S3-Compatible Blobstore</th>
<th>Duration for Internal Blobstore</th>
</tr>
</thead>
<tbody>
<tr>
<td>API unavailable</td>
<td>lock</td>
<td>15 seconds</td>
<td>15 seconds</td>
<td>15 seconds</td>
</tr>
<tr>
<td>API unavailable</td>
<td>backup</td>
<td>&lt;30 seconds</td>
<td>Proportional to blobstore size</td>
<td>10 seconds</td>
</tr>
<tr>
<td>API unavailable</td>
<td>unlock</td>
<td>3 minutes</td>
<td>3 minutes</td>
<td>3 minutes</td>
</tr>
<tr>
<td>API available</td>
<td>drain and checksum</td>
<td>&lt;10 seconds</td>
<td>&lt;10 seconds</td>
<td>Proportional to blobstore size</td>
</tr>
</tbody>
</table>

Blobstore backup and restore

Blobstores can be very large. To minimize downtime, BBR only takes blob metadata during the backup. For example, in the case of internal blobstores (Webdav/NFS), BBR takes a list of hard links that point to the blobs. Once the API becomes available, BBR makes copies of the blobs.

Unsupported Products

- **Data services.** The Pivotal data services listed below do not support BBR. Operators of these services should use the automatic backups feature of each tile, available within Ops Manager.
  - MySQL for PCF
  - Pivotal Cloud Cache for PCF
  - RabbitMQ for PCF
  - Redis for PCF
- **External blobstores and databases.** BBR support for backing up and restoring external databases and blobstores varies across PCF versions. For more information, see Supported Components and External Storage Support Across PCF Versions in Backing up Pivotal Cloud Foundry with BBR.

Best Practices

Frequency of Backups

Pivotal recommends that you take backups in proportion to the rate of change of the data in PCF to minimize the number of changes lost if a restore is required. We suggest starting with backing up every 24 hours. If app developers make frequent changes, you should increase the frequency of backups.

Retention of Backup Artifacts

Operators should retain backup artifacts based on the timeframe they need to be able to restore to. For example, if backups are taken every 24 hours and
PCF must be able to be restored to three days prior, three sets of backup artifacts should be retained.

Artifacts should be stored in two data centers other than the PCF data center. When deciding the restore timeframe, you should take other factors such as compliance and audit-ability into account.

Security

Pivotal strongly recommends that you encrypt artifacts and stored them securely.

Disaster Recovery by Recreating the Deployment

An alternative strategy for recovering PCF after a disaster is to have automation in place so that all the data can be recreated. This requires that every modification to PCF settings and state be automated, typically through use of a pipeline.

Recovery steps include creating a new PCF, recreating orgs, spaces, users, services, service bindings and other state, and re-pushing apps.

For more information about this approach, see the following Cloud Foundry Summit presentation: Multi-DC Cloud Foundry: What, Why and How?

Disaster Recovery for Different Topologies

Active-Active

To prevent app downtime, some Pivotal customers run active-active, where they run two or more identical PCF deployments in different data centers. If one PCF deployment becomes unavailable, traffic is seamlessly routed to the other deployment. To achieve identical deployments, all operations to PCF are automated so they can be applied to both PCF deployments in parallel.

Because all operations have been automated, the automation approach to disaster recovery is a viable option for active-active. Disaster recovery requires recreating PCF, then running all the automation to recreate state.

This option requires discipline to automate all changes to PCF. Some of the operations that need to be automated are the following:

- App push, re-stage, scale
- Org, space, and user create, read, update, and delete (CRUD)
- Service instance CRUD
- Service bindings CRUD
- Routes CRUD
- Security groups CRUD
- Quota CRUD

Human-initiated changes always make their way into the system. These changes can include quotas being raised, new settings being enabled, and incident responses. For this reason, Pivotal recommends taking backups even when using an automated disaster recovery strategy.

Using BBR Backup and Restore versus Recreating a Failed PCF Deployment in Active-Active

<table>
<thead>
<tr>
<th>Disaster Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restore the PCF Data</td>
</tr>
<tr>
<td>Recreate the PCF Data</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
</tr>
<tr>
<td>IaaS prepared for PCF install</td>
</tr>
<tr>
<td>IaaS prepared for PCF install</td>
</tr>
<tr>
<td><strong>Steps</strong></td>
</tr>
<tr>
<td>1. Recreate PCF</td>
</tr>
<tr>
<td>2. Restore</td>
</tr>
<tr>
<td>3. Apply changes to make restored PCF match the other active PCF</td>
</tr>
<tr>
<td>1. Recreate PCF</td>
</tr>
<tr>
<td>2. Trigger automation to recreate orgs, spaces, etc.</td>
</tr>
<tr>
<td>3. Notify app developers to re-push apps, recreate service instances and bindings</td>
</tr>
<tr>
<td><strong>RTO (Recovery Time Objective)</strong></td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
### Active-Passive

Instead of having a true active-active deployment across all layers, some Pivotal customers prefer to install a PCF or PAS deployment on a backup site. The backup site resides on-premises, in a co-location facility, or the public cloud. The backup site includes an operational deployment, with only the most critical apps ready to accept traffic should a failure occur in the primary data center. Disaster recovery in this scenario involves the following:

1. Switching traffic to the passive PCF, making it active.
2. Recovering the formerly-active PCF. Operators can choose to do this through automation, if that option is available, or by using BBR and the restore process.

The RTO and RPO for recreating the active PCF are the same as outlined in the table above.

### Reducing RTO

Both the restore and recreate data disaster recovery options require standing up a new PCF, which can take hours. If you require shorter RTO, several options involving a pre-created standby hardware and PCF are available:

<table>
<thead>
<tr>
<th>Active-cold</th>
<th>Public cloud environment ready for PCF installation, no PCF installed. This saves both IaaS costs and PCF instance costs. For on-on-premise installations, this requires hardware on standby, ready to install on, which may not be a realistic option.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active-warm</td>
<td>PCF installed on standby hardware and kept up to date, VMs scaled down to zero (spin them up each time there is a platform update), no apps installed, no orgs or spaces defined.</td>
</tr>
<tr>
<td>Active-inflate platform</td>
<td>Bare minimum PCF install, either with no applications, or a small number of each app in a stopped state. On recovery, push a small number of apps or start current apps, while simultaneously triggering automation to scale the platform to the primary node size, or a smaller version if large percentages of loss are acceptable. This mode allows you to start sending some traffic immediately, while not paying for a full non-primary platform. This method requires data seeded, but it is usually acceptable to complete data sync while platform is scaling up.</td>
</tr>
<tr>
<td>Active-inflate apps</td>
<td>Non-primary deployment scaled to the primary node size, or smaller version if large percentages of loss are acceptable, with a small number of Diego cells (VMs). On fail-over, scale Diego cells up to primary node counts. This mode allows you to start sending most traffic immediately, while not paying for all the AIs of a fully fledged node. This method requires data to be there very quickly after failure. It does not require real-time sync, but near-real time.</td>
</tr>
</tbody>
</table>

There is a trade-off between cost and RTO: the less the replacement PCF needs to be deployed and scaled, the faster the restore.

### Automating Backups

BBR generates the backup artifacts required for PCF, but does not handle scheduling, artifact management, or encryption. The BBR team has created a [starter Concourse pipeline](#) to automate backups with BBR.

Also, Stark & Wayne’s Shield can be used as a front end management tool using the BBR plugin.

### Validating Backups

To ensure that backup artifacts are valid, the BBR tool creates checksums of the generated backup artifacts, and ensures that the checksums match the
artifacts on the jumpbox.

However, the only way to be sure that the backup artifact can be used to successfully recreate PCF is to test it in the restore process. This is a cumbersome, dangerous process so should be done with care. For instructions, see Step 11: (Optional) Validate Your Backup of the Backing Up Pivotal Cloud Foundry with BBR.
Backing Up Pivotal Cloud Foundry with BBR

This topic describes the procedure for backing up your critical backend Pivotal Cloud Foundry (PCF) components with BOSH Backup and Restore (BBR), a command-line tool for backing up and restoring BOSH deployments. To restore your backup, see the Restoring Pivotal Cloud Foundry from Backup with BBR topic.

To view the BBR release notes, see the BOSH Backup and Restore Release Notes.

During the backup, BBR stops the Cloud Controller API and the Cloud Controller workers to create a consistent backup. Only the API functionality, like pushing applications or using the Cloud Foundry Command Line Interface (cf CLI) are affected. The deployed applications do not experience downtime.

**warning**
- **Security:** Backup artifacts can contain secrets. Secure backup artifacts by using encryption or other means.
- **Compatibility:** A BBR backup can only be restored to environments matching its source environment. To ensure that a restore environment is compatible with a backup see Compatibility of Restore.
- **Failed Restore:** Restore is a destructive operation. BBR is designed to restore PCF after a disaster. If a restore fails, the environment might be left in an unusable state and require re-provisioning. For more information, see Restoring Pivotal Cloud Foundry from Backup with BBR.
- **Service Data:** BBR does not back up any service data.
- **API Outage:** The Cloud Controller API does not send or receive calls during PCF restoration.

**Recommendations**

Pivotal recommends the following:

- Follow the full procedure documented in this topic when creating a backup. This ensures that you always have a consistent backup of Ops Manager and PAS to restore from.
- Back up frequently, especially before making any changes to your PCF deployment, such as the configuration of any tiles in Ops Manager.

**Supported Components**

BBR is a binary that can back up and restore BOSH deployments and BOSH Directors. BBR requires that the backup targets supply scripts that implement the backup and restore functions. BBR can communicate securely with external blobstores and databases, using TLS, if these are configured accordingly.

BBR can back up the following components:

- **Pivotal Application Service (PAS)**
- **BOSH Director**

**PAS**

BBR can back up and restore PAS configured with the following:

- An internal MySQL database or a supported external database. For a list of supported external databases, see the Supported External Databases section of Configuring Cloud Foundry for BOSH Backup and Restore in the Cloud Foundry documentation.
- An internal WebDAV/NFS blobstore, an external Amazon S3 or S3-compatible blobstore, or an external Azure blobstore.

For guidance about backing up unsupported databases and blobstores, see Unsupported External Blobstores and Databases in PAS.

**BOSH Director**

BBR can back up and restore the BOSH Director configured with the following:

- An internal PostgreSQL database or a supported external database. As part of backing up the BOSH Director, BBR backs up the BOSH UAA database and the CredHub database. For a list of supported external databases, see the Supported External Databases section of Configuring Cloud Foundry for BOSH Backup and Restore in the Cloud Foundry documentation.
An internal blobstore or an external versioned S3 blobstore.

Note: BBR support for backing up and restoring external databases and blobstores varies across PCF versions. For more information, see External Storage Support Across PCF Versions.

External Storage Support Across PCF Versions

The following table shows what types of external databases and blobstores you can back up and restore with BBR.

<table>
<thead>
<tr>
<th>Deployment Type</th>
<th>Ops Manager Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAS with External Database</td>
<td>No  No  No  Yes  Yes  Yes</td>
</tr>
<tr>
<td>BOSH Director with External Database</td>
<td>No  No  No  No  No  Yes</td>
</tr>
<tr>
<td>PAS with Amazon S3 Blobstore*</td>
<td>Add-on Add-on Add-on Yes  Yes  Yes</td>
</tr>
<tr>
<td>BOSH Director with Amazon S3 Blobstore*</td>
<td>No  No  No  No  No  No</td>
</tr>
<tr>
<td>PAS with S3-Compatible Blobstore</td>
<td>Add-on Add-on Add-on Add-on Yes  Yes  Yes</td>
</tr>
<tr>
<td>BOSH Director with S3-Compatible Blobstore</td>
<td>No  No  No  No  No  No</td>
</tr>
<tr>
<td>PAS with Azure Blobstore</td>
<td>Add-on Add-on Add-on Add-on Add-on Yes  Yes  Yes</td>
</tr>
<tr>
<td>PAS with GCS Blobstore</td>
<td>No  No  No  No  No  No</td>
</tr>
<tr>
<td>BOSH Director with GCS Blobstore</td>
<td>No  No  No  No  No  No</td>
</tr>
</tbody>
</table>

* Any S3 clone that supports the versioning API.

Unsupported External Blobstores and Databases in PAS

If you configured an unsupported external blobstore or an unsupported external database in PAS, see the following guidelines to perform a backup successfully:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>You configured a supported database and an unsupported external blobstore in PAS.</td>
<td>Follow the PCF backup process and copy the external blobstore by using the IaaS-specific tool.</td>
</tr>
<tr>
<td>You configured an unsupported external database and a supported blobstore in PAS.</td>
<td>Follow the PCF backup process and copy the external database by using the IaaS-specific tool.</td>
</tr>
<tr>
<td>You configured both an unsupported external database and an unsupported external blobstore in PAS.</td>
<td>Follow the PCF backup process but skip Back Up Your PAS Deployment. Copy the external database and blobstore by using the IaaS-specific tool.</td>
</tr>
</tbody>
</table>

⚠️ warning: You might encounter inconsistencies between the blobstore and database. If apps do not appear during a restore, re-push those apps.

Backing Up Services

⚠️ warning: BBR does not currently back up any service data.

Keep in mind the following when backing up services:

- BBR backs up and restores the service bindings, but not the service data.
- You can redeploy on-demand service instances manually during restore, but the data in the instance is not backed up.
- BBR does not back up managed services or their data.
- You can back up and restore brokered services with the procedures documented in this topic and in the Restoring Pivotal Cloud Foundry from Backup with BBR topic.
Backup Workflow

BBR examines the jobs in the BOSH deployment, and triggers the scripts in the following stages:

1. **Pre-backup lock**: The pre-backup lock scripts locks the job so backups are consistent across the cluster.

2. **Backup**: The backup script backs up the release.

3. **Post-backup unlock**: The post-backup unlock script unlocks the job after the backup is complete.

Scripts in the same stage are all triggered together. For instance, BBR triggers all pre-backup lock scripts before any backup scripts. Scripts within a stage can be triggered in any order.

The backup artifacts are drained to the jumpbox, where the operator can transfer them to storage and use them to restore PCF.

The following diagram shows a sample backup flow.

![bbr Backup Workflow Diagram](image)

Preparing to Create Your Backup

**Step 1: Set Up Your Jumpbox**

You must have a jumpbox before you can install BBR to the jumpbox. A jumpbox is a separate, hardened server on your network that provides a controlled means of access to the VMs other computers on your network.

Pivotal recommends using the Ops Manager VM as your jumpbox. If you use the Ops Manager VM as your jumpbox, the path to the root Certificate Authority (CA) certificate is:

```
/var/temp/workspaces/default/root_ca_certificate
```

You must have this path to run BBR commands.
As an alternative to using the Ops Manager VM as your jumpbox, you can do the following:

- Use Concourse to back up a PCF installation using BBR. For more information, see the [BBR PCF Pipeline Tasks GitHub repo.](#)
- Configure a custom jumpbox. For more information, see [Configuring a custom jumpbox for BBR](#) in the Pivotal Support Knowledge Base.

**Step 2: Record the Cloud Controller Database Encryption Credentials**

Perform the following steps to retrieve the Cloud Controller Database encryption credentials from the PAS tile:

1. Navigate to Ops Manager in a browser and log in to the Ops Manager Installation Dashboard.
2. Select PAS **Credentials** and locate the Cloud Controller section.
3. Record the **Db Encryption Credentials**. You must provide these credentials when you contact [Pivotal Support](#) for assistance restoring your installation.

<table>
<thead>
<tr>
<th>Cloud Controller</th>
<th>VM Credentials</th>
<th>Link to Credential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Db Encryption Credentials</td>
<td></td>
<td>Link to Credential</td>
</tr>
<tr>
<td>Encrypt Key</td>
<td></td>
<td>Link to Credential</td>
</tr>
<tr>
<td>Locket Client Cert</td>
<td></td>
<td>Link to Credential</td>
</tr>
</tbody>
</table>

**Step 3: Retrieve BOSH Director Credentials**

To use BBR, you must retrieve and record the following credentials:

- Bosh Director Credentials
- Bbr Ssh Credentials
- Uaa Bbr Client Credentials

There are two ways to retrieve BOSH Director credentials:

- **Ops Manager Installation Dashboard**
- **Ops Manager API**

**Option 1: Ops Manager Installation Dashboard**

To retrieve your BOSH Director credentials using the Ops Manager Installation Dashboard, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Click the **Credentials** tab.
4. Locate **Director Credentials**.
   a. Click **Link to Credentials** next to it.
   b. Verify the value of the `identity` field. It should be `director`.
   c. Copy and record the value of the `password` field.

5. Locate **Bbr Ssh Credentials**.
   a. Click **Link to Credentials** next to it.
   b. Copy the value of the `private_key_pem` field.
   c. Run the following command to reformat the copied value, and save it in the current directory to a file named **PRIVATE-KEY-FILE**:

   ```
   printf "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
   ```

   Where:
   - `YOUR-PRIVATE-KEY` is the text of your private key.
   - `PRIVATE-KEY-FILE` is the path to the private key file you are creating.

   For example:

   ```
   $ printf "-----BEGIN RSA PRIVATE KEY-----\nMIIEkeycontents END RSA PRIVATE KEY-----" > bbr_key.pem
   ```

6. Locate the **Uaa Bbr Client Credentials**
   a. Click **Link to Credentials** next to it.
b. Verify the value of the `identity` field. It should be `bbr_client`.

c. Record the value of the `password` field.

**Option 2: Ops Manager API**

To retrieve BOSH Director credentials using the Ops Manager API, perform the following steps:

1. Obtain your UAA access token. For more information, see Access the API.

2. Retrieve the **Director Credentials** by performing the following steps:

   a. Run the following command:

   ```
curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/director_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Where:
   - **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - **UAA-ACCESS-TOKEN** is your UAA access token.

   b. Verify the value of the `identity` field. It should be `director`.

   c. Record the value of the `password` field.

3. Retrieve the **Bbr Ssh Credentials** by performing the following steps:

   a. Run the following command:

   ```
curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Where:
   - **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - **UAA-ACCESS-TOKEN** is your UAA access token.

   b. Copy the value of the `private_key_pem` field.

   c. Run the following command to reformat the copied value, and save it in the current directory to a file named `PRIVATE-KEY-FILE`:

   ```
printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
   ```

   Where:
   - **YOUR-PRIVATE-KEY** is the text of your private key.
   - **PRIVATE-KEY-FILE** is the path to the private key file you are creating.

   For example:

   ```
$ printf -- "-----BEGIN RSA PRIVATE KEY-----
MIIEkeycontents
-----END RSA PRIVATE KEY-----" > bbr_key.pem
   ```

4. Retrieve the **Uaa Bbr Client Credentials** by performing the following steps:

   a. Run the following command:

   ```
curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/uaa_bbr_client_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ```

   Where:
   - **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - **UAA-ACCESS-TOKEN** is your UAA access token.

   b. Verify the value of the `identity` field. It should be `bbr_client`.

   c. Record the value of the `password` field.

For more information about using the Ops Manager API, see the Using the Ops Manager API topic.
Step 4: Retrieve BOSH Director Address

Perform the following steps to retrieve the IP address of your BOSH Director from the BOSH Director tile:

1. If you are not using the Ops Manager VM as your jumpbox, install the latest BOSH CLI on your jumpbox.

2. From the Installation Dashboard in Ops Manager, select **BOSH Director > Status** and record the IP address listed for the Director. You access the BOSH Director using this IP address.

3. From the command line, log into the BOSH Director, using the IP address that you recorded above, by running the following command:

   ```
   bosh -e DIRECTOR-IP \
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE log-in
   ```

   Where:
   - **DIRECTOR-IP** is the BOSH Director IP address recorded above.
   - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the root Certificate Authority (CA) certificate as outlined in **Step 1: Set Up your Jumpbox**, above.

4. When prompted for **Email ()**, specify **director**.

5. When prompted for **Password ()**, enter the **Director Credentials** that you obtained in **Retrieve BOSH Director Credentials**.
   For example:
   ```
   $ bosh -e 10.0.0.3 \
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate log-in
   Email (): director
   Password (): *******************
   Successfully authenticated with UAA
   Succeeded
   ```

Step 5: Check Your BOSH Director

Perform the following steps to confirm that your BOSH Director is reachable and can be backed up.

Connect to Your Jumpbox

You can establish a connection to your jumpbox in one of the following ways.

- **Connect with SSH**
- **Connect with BOSH_ALL_PROXY**

**Note:** These procedures require you have a jumpbox configured for SSH access. If you do not have a jumpbox, please see **Enabling SSH Access** and **Tunneling** in the BOSH documentation.

Connect with SSH

There are two options available to you to connect to your jumpbox with SSH:
• Ops Manager VM.
• SSH via command line.

1. To connect to your jumpbox via Ops Manager VM, perform the following steps:
   a. Log in to the Ops Manager VM:
   b. If you are using the Ops Manager VM as your jumpbox, see the Log in to the Ops Manager VM with SSH section of Advanced Troubleshooting with the BOSH CLI for information on how to use SSH to connect to the Ops Manager VM.

2. To connect to your jumpbox via command line, perform the following steps:
   a. Run the following command to SSH into your jumpbox:
   ```bash
   ssh -i LOCAL-PATH-TO-JUMPBOX-PRIVATE-KEY JUMPBOX-USER@JUMPBOX-ADDRESS
   ```
   Where:
   - `LOCAL-PATH-TO-JUMPBOX-PRIVATE-KEY` is the local path to your private key file for the jumpbox host.
   - `JUMPBOX-USER` is your jumpbox username.
   - `JUMPBOX-ADDRESS` is the IP address of your jumpbox.

   **Note:** If you connect to your jumpbox with SSH, you must run the BBR commands in the following sections from within your jumpbox.

   **Connect with BOSH_ALL_PROXY**

   Set and use `BOSH_ALL_PROXY`. Using `BOSH_ALL_PROXY` opens an SSH tunnel with SOCKS5 to the jumpbox. This tunnel enables you to forward requests to the BOSH Director through the jumpbox from your local machine.

   Use one of the following methods to create the tunnel:

   • **Tunnel created by BOSH CLI:** To provide the BOSH CLI with the SSH credentials it needs to create the tunnel, run the following command:
   ```bash
   export BOSH_ALL_PROXY=ssh+socks5://JUMPBOX@JUMPBOX-IP:SOCKS-PORT?private_key=JUMPBOX-KEY-FILE
   ```
   Where:
   - `JUMPBOX` is the name of your jumpbox.
   - `JUMPBOX-IP` is the IP address of the jumpbox.
   - `SOCKS-PORT` is the local SOCKS port.
   - `JUMPBOX-KEY-FILE` is the local SSH private key for accessing the jumpbox.

   • **Tunnel established separately:**

   1. To establish the tunnel and make it available on a local port, run the following command:
   ```bash
   ssh -4 -D SOCKS-PORT -i JUMPBOX-KEY-FILE JUMPBOX@JUMPBOX-IP
   ```
   Where:
   - `SOCKS-PORT` is the local SOCKS port.
   - `JUMPBOX` is the name of your jumpbox.
   - `JUMPBOX-IP` is the IP address of the jumpbox.
   - `JUMPBOX-KEY-FILE` is the local SSH private key for accessing the jumpbox.

   2. To provide the BOSH CLI with access to the tunnel through use of the `BOSH_ALL_PROXY` environment variable, run the following command:
   ```bash
   export BOSH_ALL_PROXY=socks5://localhost:12345
   ```
   **Note:** Ensure the Socks port is not already in use by a different tunnel/process.

   **Note:** Using `BOSH_ALL_PROXY` can result in longer backup and restore times due to network performance degradation. Because all operations must pass through the proxy, moving backup artifacts can be significantly slower.
Check Your BOSH Director

1. Run the BBR `pre-backup-check` command:

   ```
   bbr director \
   --private-key-path PRIVATE-KEY-FILE \
   --username bbr \
   --host HOST \
   pre-backup-check
   ```

   Where:
   - `PRIVATE-KEY-FILE` is the path to the private key file that you created from Bbr Ssh Credentials in Step 3: Retrieve BOSH Director Credentials.
   - `HOST` is the address of the BOSH Director. If the BOSH Director is public, `HOST` is a URL, such as `https://my-bosh.xxx.cf-app.com`.
   - If the BOSH Director is on an internal network, `HOST` is the BOSH-DIRECTOR-IP which you retrieved in the Step 4: Retrieve BOSH Director Address section of this topic.

   **Note:** Use the optional `--debug` flag to enable debug logs. See the Logging section of this topic for more information.

2. After the pre-backup check succeeds, continue to Step 5: Confirm Backup Restore Node is Deployed below.

   If the pre-backup check fails, the BOSH Director might not have the correct backup scripts, or the connection to the BOSH Director might have failed.

Step 6: Confirm Backup Restore Node is Deployed

When BBR backs up PAS, it needs a Backup Restore node. This procedure confirms that the Backup Restore node is deployed.

1. In Ops Manager, open the Pivotal Application Service (PAS) tile.

2. In the Resource Config pane, find the Backup Restore Node job and check if the Instances dropdown is set to `1`.

   ![Backup Restore Node](image)

3. If the Instances dropdown is not set to `1`:
   a. Set the Instances dropdown to `1`.
   b. Click Save.
   c. Navigate to the Ops Manager Installation Dashboard.
   d. Click Review Pending Changes.
   e. Review your changes. For more information, see Reviewing Pending Product Changes.
   f. Click Apply Changes.

Step 7: Identify Your Deployment

1. Log in to your BOSH Director.

2. To identify the name of the BOSH deployment that contains PCF, run the following command:

   ```
   $ bosh -e BOSH-DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE deployments
   ```

   Where:
   - `BOSH-DIRECTOR-IP` is the BOSH Director IP retrieved in Step 4: Retrieve BOSH Director Address.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the Certificate Authority (CA) certificate for the BOSH Director.

   ![Deployment List](image)

   In the above example, the name of the BOSH deployment that contains PCF is `cf-example`.
Step 8: Check Your Deployment

Perform the following steps to check that your BOSH Director is reachable and has a deployment that can be backed up:

1. Run the BBR pre-backup check:

   ```
   bbr deployment
   --target BOSH-DIRECTOR-IP
   --username BOSH-CLIENT
   --password BOSH-PASSWORD
   --deployment DEPLOYMENT-NAME
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE
   pre-backup-check
   ```

   Where:
   - **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 4: Retrieve BOSH Director Address.
   - **BOSH-CLIENT** and **BOSH-PASSWORD** are the Uaa Bbr Client Credentials, identity and password, that you retrieved in Step 3: Retrieve BOSH Director Credentials.
   - **DEPLOYMENT-NAME** is the deployment name retrieved in Step 6: Identify Your Deployment.
   - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the root CA certificate for the BOSH Director. If you are using the Ops Manager VM as your jumpbox, the path is `/var/tempest/workspaces/default/root_ca_certificate`.

2. After the pre-backup check succeeds, continue to Create Your Backup below.

   If the pre-backup check fails, the deployment you selected might not have the correct backup scripts, or the connection to the BOSH Director might have failed.

Creating Your Backup

Step 9: Export Installation Settings

Pivotal recommends that you back up your Ops Manager installation settings by exporting frequently.

When exporting your installation settings, keep in mind the following:

- This option is only available after you have deployed at least once.
- Always export your installation settings before following the steps in the Deploy Ops Manager and Import Installation Settings section of the Restoring Pivotal Cloud Foundry from Backup with BBR topic.
- Exporting your installation settings only backs up the settings you configured in Ops Manager. It does not back up your VMs or any external MySQL databases.
- Your Ops Manager settings are encrypted. Make sure you keep track of your Decryption Passphrase because this is needed to restore the Ops Manager settings.

**Note:** When exporting installation settings for Ops Manager v1.12 and later, releases are not included in the output file. Releases are now included in the BOSH Director backup, therefore, the output file is much smaller than in previous Ops Manager versions.

There are two ways to export Ops Manager installation settings:

- **Use the Ops Manager UI**
- **Use the Ops Manager API**

Option 1: Use the Ops Manager UI

To export your installation settings using the Ops Manager UI, do the following:

1. From the **Installation Dashboard** in the Ops Manager interface, click your username at the top right navigation.
2. Select **Settings**.
3. Select **Export Installation Settings**.
4. Click **Export Installation Settings**.

Option 2: Use the Ops Manager API

If you want to automate the backup process, you can use the Ops Manager API to export your installation settings.

1. To export your installation settings using the Ops Manager API, run the following command:

   ```bash
   curl https://OPS-MAN-FQDN/api/v0/installation_asset_collection 
   -H "Authorization: Bearer UAA-ACCESS-TOKEN" > installation.zip
   ```

   Where:
   - **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - **UAA-ACCESS-TOKEN** is your UAA access token. For more information, see [Access the API](#).

Step 10: Back Up Your BOSH Director

**Notes:**
- The BOSH Director backup can take more than 10 minutes.
- Because the BBR **backup** command can take a long time to complete, Pivotal recommends you run it independently of the SSH session so that the process can continue running even if your connection to the jumpbox fails.

Use `nohup`, a `screen`, or a `tmux` session.

1. Run the BBR **backup** command to back up your BOSH Director:

   ```bash
   bbr director 
   --private-key-path PRIVATE-KEY-FILE 
   --username bbr 
   --host HOST 
   backup
   ```

   Where:
   - **PRIVATE-KEY-FILE** is the path to the private key file that you created from **Bbr Ssh Credentials** in [Step 3: Retrieve BOSH Director Credentials](#).
   - **HOST** is the address of the BOSH Director:
     - If the BOSH Director is public, **HOST** is a URL, such as: `https://my-bosh.xxx.cf-app.com`.
     - If the BOSH Director is not public, **HOST** is the **BOSS-DIRECTOR-IP** retrieved in [Step 4: Retrieve BOSH Director Address](#).
If your backup terminates early or fails you will need to perform a clean up.

1. To clean up, run the following `backup-cleanup` command:

   ```bash
   bbr director
   --private-key-path PRIVATE-KEY-FILE
   --username bbr
   --host HOST
   backup-cleanup
   ```

   Where:
   - `PRIVATE-KEY-FILE` is the path to the private key file that you created from Bbr Ssh Credentials in Step 3: Retrieve BOSH Director Credentials.
   - `HOST` is the address of the BOSH Director:
     - If the BOSH Director is public, `HOST` is a URL, such as `https://my-bosh.xxx.cf-app.com`.
     - If the BOSH Director is not public, `HOST` is the BOSH-DIRECTOR-IP retrieved in Step 4: Retrieve BOSH Director Address.

---

**Step 11: Back Up Your PAS Deployment**

**Notes:**
- Backing up PAS can take more than 5 minutes, and can take considerably longer with larger blobstores or slow network connections. The backup also incurs Cloud Controller downtime during which users are unable to push, scale, or delete apps. Your apps will not be affected.
- Because the BBR `backup` command can take a long time to complete, Pivotal recommends you run it independently of the SSH session, so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup` but you could also run the command in a `screen` or `tmux` session.

1. (Optional) If you are not using an internal blobstore or a versioned S3-compatible external blobstore, create a copy of the blobstore with your IaaS specific tool. Your blobstore backup might be slightly inconsistent with your PAS backup depending on the duration of time between performing the backups.

2. Run the BBR `backup` command to back up your PAS deployment:

   ```bash
   bbr deployment
   --target BOSH-DIRECTOR-IP
   --username BOSH-CLIENT
   --password BOSH-PASSWORD
   --deployment DEPLOYMENT-NAME
   --ca-cert PATH-TO-BOSH-SERVER-Certificate
   backup
   ```

   Where:
   - `BOSH-DIRECTOR-IP` is the BOSH Director IP retrieved in Step 4: Retrieve BOSH Director Address.
   - `BOSH-CLIENT`, `BOSH-PASSWORD` are the Uaa Bbr Client Credentials, identity and password, that you retrieved in Step 3: Retrieve BOSH Director Credentials.
   - `DEPLOYMENT-NAME` is the deployment name retrieved in Step 6: Identify Your Deployment.
   - `PATH-TO-BOSH-SERVER-Certificate` is the path to the Certificate Authority (CA) certificate for the BOSH Director.
   - Use the optional `--debug` flag to enable debug logs. For more information, see Logging below.
   - Use the optional `--with-manifest` flag to ensure that BBR downloads your current deployment manifest when backing up. These manifests are included in the Ops Manager export, but are useful for reference.

3. After the command successfully completes, continue to Step 11: Transfer Backup Artifacts.

   If the command fails, perform the following steps:
   a. Run the BBR `backup-cleanup` command:

---

© Copyright Pivotal Software Inc, 2013-2019
bbr deployment
  --target BOSH-DIRECTOR-IP
  --username BOSH-CLIENT
  --password BOSH-PASSWORD
  --deployment DEPLOYMENT-NAME
  --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE
backup-cleanup

Where:

- **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 4: Retrieve BOSH Director Address.
- **BOSH-CLIENT**, **BOSH-PASSWORD** are the Uaa Bbr Client Credentials, identity and password, that you retrieved in Step 3: Retrieve BOSH Director Credentials.
- **DEPLOYMENT-NAME** is the deployment name retrieved in Step 6: Identify Your Deployment.
- **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director.
- Use the optional **--debug** flag to enable debug logs. For more information, see Logging below.
- Use the optional **--with-manifest** flag to ensure that BBR downloads your current deployment manifest when backing up. These manifests are included in the Ops Manager export, but are useful for reference.

b. Run the BBR `backup` command again after checking the following:

- All the parameters in the command are set.
- The BOSH Director credentials are valid.
- The specified deployment exists.

After Taking Your Backup

**Step 12: Transfer Backup Artifacts**

After creating your backup successfully, perform the following steps:

1. Move the backup artifacts off the jumpbox to your preferred storage space. The backup created by BBR consists of a folder with the backup artifacts and metadata files. Pivotal recommends compressing and encrypting the files.

2. Make redundant copies of your backup artifacts and store them in multiple locations to minimize the risk of losing backups in the event of a disaster.

3. Attempt to restore every backup to validate the artifacts. Perform the procedures in the next step, Step 12: Validate Your Backup.

Note: Backup artifacts might contain data covered by the European Union’s General Data Protection Regulation (GDPR).

**Step 13: (Optional) Validate Your Backup**

After backing up PCF, consider validating your backup by restoring it and checking the applications. BBR is designed for disaster recovery and BBR backups are only compatible with environments matching their source environment’s configuration.

⚠️ warning:
- The VMs and disks from the backed-up BOSH Director must not be visible to the new BOSH Director when validating your backup. As a result, Pivotal recommends that you deploy the new BOSH Director to a different IaaS network and account than the VMs and disks of the backed-up BOSH Director.
- Data services outside of PCF might produce unexpected side effects during restoration. Restored apps and services might attempt to connect to data services when you restore to a new environment. For example, consider an app that processes mail queues and connects to an external database. When you validate your backup in a test environment, the app might start processing the queue, and this work might be lost.

Validate Your Backup in a Second Environment

To spin up a second environment and test the backup, perform the following steps:
1. Export your Ops Manager installation by performing the steps in the [Step 8: Export Installation Settings](#) section above.

2. Create a new Ops Manager VM in a different network to the original. Ensure that the Ops Manager VM has enough persistent disk to accommodate the files exported in the previous step. Consult the topic for [deploying PCF on your IaaS](#).

3. Ensure that the restore environment is compatible with the backup. For more information, see [Compatibility of Restore](#).

4. Follow the instructions in the [Restoring Pivotal Cloud Foundry from Backup with BBR](#) topic.

**Validate Your PAS Backup Only**

For a sandbox or other non-production environment, you can optionally perform an in-place restore of PAS only. In this case, you restore the PAS backup to the same PCF environment that the backup was created from. Follow the procedures in [Restoring PCF from Backup with BBR](#).

**Logging**

BBR outputs logs to stdout. By default, BBR logs:

- The backup and restore scripts that it finds
- When it starts or finishes a stage, such as pre-backup scripts or backup scripts
- When the process is complete
- When any error occurs

If you require additional logging, use the optional `--debug` flag to print the following information:

- Logs about the API requests made to the BOSH server
- All commands executed on remote instances
- All commands executed on local environment
- Standard in and standard out streams for the backup and restore scripts when they are executed

**Canceling a Backup**

If you need to cancel a backup, perform the following steps:

1. Enter `CTRL-C` to terminate the BBR process, then enter `yes` to confirm.

2. Run the BBR `backup-cleanup` command.

   - For a canceled director backup, run the following:

     ```bash
     bbr director
     --private-key-path PRIVATE-KEY-FILE
     --username bbr
     --host HOST
     backup-cleanup
     ```

     Where:

     - `PRIVATE-KEY-FILE` is the path to the private key file that you created from Bbr Ssh Credentials in [Step 3: Retrieve BOSH Director Credentials](#).
     - `HOST` is the address of the BOSH Director:

       - If the BOSH Director is public, `HOST` is a URL, such as `https://my-bosh.xxx.cf-app.com`.
       - If the BOSH Director is not public, `HOST` is the `BOSH-DIRECTOR-IP` retrieved in [Step 4: Retrieve BOSH Director Address](#).

   - For a canceled deployment backup, run the following:

     ```bash
     bbr deployment
     --target BOSH-DIRECTOR-IP
     --username BOSH-CLIENT
     --password BOSH-PASSWORD
     --deployment DEPLOYMENT-NAME
     --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE
     backup-cleanup
     ```
Where:

- **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in [Step 4: Retrieve BOSH Director Address](#).
- **BOSH-CLIENT**, **BOSH-PASSWORD** are the **Uaa Bbr Client Credentials**, identity and password, that you retrieved in [Step 3: Retrieve BOSH Director Credentials](#).
- **DEPLOYMENT-NAME** is the deployment name retrieved in [Step 6: Identify Your Deployment](#).
- **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director.
- Use the optional **--debug** flag to enable debug logs. For more information, see **Logging** below.
- Use the optional **--with-manifest** flag to ensure that BBR downloads your current deployment manifest when backing up. These manifests are included in the Ops Manager export, but are useful for reference.

---

**European Union’s General Data Protection Regulation (GDPR)**

Backup artifacts might contain personal and other data covered by the European Union’s [General Data Protection Regulation (GDPR)](https://www.gov.uk/government/publications/gdpr-technical-terms). For example, a PAS backup could contain user email addresses. As such, handle backup artifacts in accordance with organizational policies and applicable laws as they pertain to security, confidentiality, and privacy.
PAS Component Availability During Backup

This topic describes the operational impact of backing up Pivotal Application Service (PAS) components with BBR. To ensure correctness of backups, each component that requires backup has its own set of scripts. The sections in this topic describe the availability of each component during backup of its database.

Cloud Controller

The Cloud Controller is unavailable during backup. Apps and Services continue to run as normal, but you cannot perform operations that require the Cloud Controller API. This includes the following:

- Pushing new apps or creating new services
- Modifying existing apps or services
- Using the clients of the Cloud Controller API, such as the following:
  - The Cloud Foundry Command Line Interface (cf CLI)
  - Apps Manager and its integrations
  - The Java client used by Spring apps

The backup process for the Cloud Controller is as follows:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Pre-backup lock</td>
<td>The processes running on the Cloud Controller Worker, Cloud Controller, and Clock Global VMs are stopped.</td>
</tr>
<tr>
<td>2: Backup</td>
<td>The BBR SDK backup script runs to backup the Cloud Controller database (CCDB), which contains state information for apps on your deployment.</td>
</tr>
<tr>
<td>3: Post-backup unlock</td>
<td>The processes start again on the Cloud Controller Worker, Cloud Controller, and Clock Global VMs.</td>
</tr>
</tbody>
</table>

UAA

UAA remains available in read-only mode during backup. This means that you cannot perform write operations for clients, users, groups, identity providers, or zone configuration. However, you can continue performing read operations, such as generating, validating, and revoking tokens. Additionally, UAA continues to authenticate users and authorize requests for users and clients.

The read-only behavior during backup applies to all of the following ways of accessing UAA: the UAA API, the UAA CLI, cf CLI, login screens, and services such as the Single Sign-On Service tile.

The backup process for UAA is as follows:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Pre-backup lock</td>
<td>UAA enters read-only mode.</td>
</tr>
<tr>
<td>2: Backup</td>
<td>The BBR SDK backup script runs to backup the UAA database, which contains Cloud Foundry user credentials.</td>
</tr>
<tr>
<td>3: Post-backup unlock</td>
<td>UAA exits read-only mode.</td>
</tr>
</tbody>
</table>

Routing API

The Routing API remains available during backup. However, you cannot perform write operations using the Routing API because the routing database is locked. All read operations offered by the Routing API remain available.

The BBR SDK backup script for the Routing API backs up its database, which contains router groups, routes, and internal implementation information.

Usage Service

The Usage Service is unavailable during backup. You cannot access the API as described in Monitoring App, Task, and Service Instance Usage. Additionally, you cannot view usage and accounting reports as described in Monitoring Instance Usage with Apps Manager.
The backup process for Usage Service is as follows:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Pre-backup lock</td>
<td>The Usage Service apps in the <code>system</code> org stop. This lock occurs before the Cloud Controller and UAA components lock.</td>
</tr>
<tr>
<td>2: Backup</td>
<td>The BBR SDK backup script runs to backup the Usage Service database.</td>
</tr>
<tr>
<td>3: Post-backup unlock</td>
<td>The Usage Service apps in the <code>system</code> org start again. This unlock occurs after the Cloud Controller and UAA components unlock.</td>
</tr>
</tbody>
</table>

**App Autoscaler**

The App Autoscaler service is unavailable during backup. You cannot access the UI or API. For any apps configured to use the App Autoscaler, the service does not scale these apps during backup.

The backup process for App Autoscaler is as follows:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Pre-backup lock</td>
<td>The Autoscaler apps in the <code>system</code> org stop. This lock occurs before the Cloud Controller and UAA components lock.</td>
</tr>
<tr>
<td>2: Backup</td>
<td>The BBR SDK backup script runs to backup the App Autoscaler database.</td>
</tr>
<tr>
<td>3: Post-backup unlock</td>
<td>The Autoscaler apps in the <code>system</code> org start again. This unlock occurs after the Cloud Controller and UAA components unlock.</td>
</tr>
</tbody>
</table>

**NFS Volume Service**

The NFS service broker backup scripts rely on the locking of the Cloud Controller to stop traffic to its service. This is because the Cloud Controller is responsible for invoking the NFS service broker.

When the Cloud Controller locks during backup, you cannot create or delete new instances or bindings of a volume service. However, apps already bound to a volume service continue to operate normally during backups.

The NFS service broker backup script performs a backup of the database used to store service instances and service bindings for the NFS service broker.

**Notification Service**

The Notification Service is not available during backup with BBR due to its dependency on the Cloud Controller. Notifications cannot be sent while the Cloud Controller is unavailable.

**Network Policy Server**

The Network Policy Server is unavailable during backup. While existing policies are still enforced, you cannot use the `cf` CLI to add or remove policies for Container Networking as documented in *Administering Container-to-Container Networking*.

**CredHub**

Runtime CredHub is unavailable during backup. If the service instance credentials for an app are stored in CredHub, the app cannot fetch those credentials during backup. In some cases, apps may not start if they cannot fetch credentials for a service instance binding.
Restoring PCF from Backup with BBR

This topic describes the procedure for restoring your critical backend Pivotal Cloud Foundry (PCF) components with BOSH Backup and Restore (BBR), a command-line tool for backing up and restoring BOSH deployments. To perform the procedures in this topic, you must have backed up PCF by following the steps in the *Backing Up Pivotal Cloud Foundry with BBR* topic.

To view the BBR release notes, see [BOSH Backup and Restore Release Notes](#).

The procedures described in this topic prepare your environment for PCF, deploy Ops Manager, import your installation settings, and use BBR to restore your PCF components.

⚠️ **warning**: Restoring PCF with BBR is a destructive operation. If the restore fails, the new environment may be left in an unusable state and require re-provisioning. Only perform the procedures in this topic for the purpose of disaster recovery, such as recreating PCF after a storage-area network (SAN) corruption.

⚠️ **warning**: When validating your backup, the VMs and disks from the backed-up BOSH Director should not be visible to the new BOSH Director. As a result, Pivotal recommends that you deploy the new BOSH Director to a different IaaS network and account than the VMs and disks of the backed-up BOSH Director.

⚠️ **warning**: For PCF v2.0, BBR only supports backup and restore of environments with zero or one CredHub instances.

💡 **Note**: If the BOSH Director you are restoring had any deployments that were deployed manually rather than through an Ops Manager tile, you must restore them manually at the end of the process. For more information, see [Optional Step 16: Restore Non-Tile Deployments](#).

Compatibility of Restore

This section describes the restrictions for a backup artifact to be restorable to another environment. This section is for guidance only, and Pivotal highly recommends that operators validate their backups by using the backup artifacts in a restore.

Consult the following restrictions for a backup artifact to be restorable:

- **CIDR ranges**: BBR requires the IP address ranges to be the same in the restore environment as in the backup environment.
- **Topology**: BBR requires the BOSH topology of a deployment to be the same in the restore environment as it was in the backup environment.
- **Naming of instance groups and jobs**: For any deployment that implements the backup and restore scripts, the instance groups and jobs must have the same names.
- **Number of instance groups and jobs**: For instance groups and jobs that have backup and restore scripts, there must be the same number of instances.
- **Limited validation**: BBR puts the backed up data into the corresponding instance groups and jobs in the restored environment, but can't validate the restore beyond that. For example, if the MySQL encryption key is different in the restore environment, the BBR restore might succeed although the restored MySQL database is unusable.
- **PCF version**: BBR can restore to the same version of PCF that was backed up. BBR does not support restoring to other major, minor, or patch releases.

💡 **Note**: A change in VM size or underlying hardware will not affect BBR's ability to restore data, as long as there is adequate storage space to restore the data.

Restore Workflow

Click the diagram below to see the full-size image.
The diagram above shows the flow of the PCF restore process in a series of steps performed by the PCF operator. The following steps will be covered in more detail throughout this topic.

1. **Launch new Ops Manager:** Perform the procedures for your IaaS to deploy Ops Manager. See part one of the Deploy Ops Manager and Import Installation Settings step below for more information.

2. **Import settings:** You can import settings either with the Ops Manager UI or API. See part two of the Deploy Ops Manager and Import Installation Settings step below for more information.

3. **Remove bosh-state.json:** SSH into your Ops Manager VM and delete the `bosh-state.json` file. See the Remove BOSH State File step below for more information.

4. **Apply Changes (Director only):** Use the Ops Manager API to only deploy the BOSH Director. See the Deploy the BOSH Director step below for more information.

5. **bbr restore <director>:** Run the BBR restore command from your jumpbox to restore the BOSH Director. See the Restore the BOSH Director step below for more information.

6. **Use BOSH cck to fix the stale cloud ID references in the BOSH database:** For each deployment in the BOSH Director, you will need to run a `bosh cloud-check` command. See the Remove Stale Cloud IDs for All Deployments step for more information.

7. **Apply Changes:** On the Ops Manager Installation Dashboard, click Review Pending Changes, review your changes, and then click Apply Changes. For more information, see Reviewing Pending Product Changes.

8. **bbr restore <PAS>:** Run the BBR restore command from your jumpbox to restore PAS. See the Restore PAS step below for more information.
Prepare to Restore

This section provides the steps you need to perform before restoring your PCF backup with BBR.

Step 1: (Optional) Prepare Your Environment

In an event of a disaster, you may lose not only your VMs and disks, but your IaaS resources as well, such as networks and load balancers.

If you need to recreate your IaaS resources, prepare your environment for PCF by following the instructions specific to your IaaS in Installing Pivotal Cloud Foundry.

Note: The instructions for installing PCF on Amazon Web Services (AWS) and OpenStack combine the procedures for preparing your environment and deploying Ops Manager into a single topic. The instructions for the other supported IaaSes split these procedures into two separate topics.

If you recreate your IaaS resources, you must also add those resources to Ops Manager by performing the procedures in the Step 3: (Optional) Configure Ops Manager for New Resources section.

Step 2: Deploy Ops Manager and Import Installation Settings

1. Perform the procedures for your IaaS to deploy Ops Manager:

   - AWS:
     - If you configured AWS manually, see Step 12: Launch an Ops Manager API through Step 19: Create a MySQL Database Using AWS RDS of Installing PCF on AWS Manually.
     - If you used Terraform to install PCF on AWS, see Installing PCF on AWS Using Terraform.
   - GCP: See Deploying Ops Manager on GCP Manually.
   - OpenStack: See Step 4: Create Ops Manager Image through Step 9: Create a DNS Entry in Deploying Ops Manager to OpenStack.
   - vSphere: See Deploying Ops Manager on vSphere.

2. Import your installation settings. This can be done in two ways:

   a. Using the Ops Manager UI

      i. Access your new Ops Manager by navigating to YOUR-OPS-MAN-FQDN in a browser.
      
      ii. On the Welcome to Ops Manager page, click Import Existing Installation.

     iii. In the import panel, perform the following tasks:

        - Enter the Decryption Passphrase in use when you exported the installation settings from Ops Manager.
        - Click Choose File and browse to the installation zip file that you exported in the Step 9: Export Installation Settings section of the Backing Up Pivotal Cloud Foundry with BBR topic.
iv. Click Import.

Note: Some browsers do not provide feedback on the status of the import process, and may appear to hang. The import process takes at least 10 minutes, and takes longer the more tiles that were present on the backed-up Ops Manager.

v. A Successfully imported installation message appears upon completion.

Using the Ops Manager API:

```bash
curl "https://OPS-MAN-FQDN/api/v1/installation_asset_collection" 
-X POST 
-H "Authorization: Bearer UAA-ACCESS-TOKEN" 
-F 'installation[file]=@installation.zip' 
-F 'passphrase=DECRYPTION-PASSPHRASE'
```

Where:
- **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
- **UAA-ACCESS-TOKEN** is the UAA access token. For more information about how to retrieve this token, see Using the Ops Manager API.
- **DECRYPTION-PASSPHRASE** is the decryption passphrase in use when you exported the installation settings from Ops Manager.

⚠️ warning: Do not click Apply Changes in Ops Manager until the instruction in Step 14: Redeploy PAS.

Step 3: (Optional) Configure Ops Manager for New Resources

If you recreated IaaS resources such as networks and load balancers by following the steps in the Step 1: (Optional) Prepare Your Environment section above, perform the following steps to update Ops Manager with your new resources:

1. Enable Ops Manager advanced mode. For more information, see How to Enable Advanced Mode in the Ops Manager in the Pivotal Knowledge Base.

   Note: In advanced mode Ops Manager will allow you to make changes that are normally disabled. You may see warning messages when you save changes.

2. Navigate to the Ops Manager Installation Dashboard and click the BOSH Director tile.

3. If you are using Google Cloud Platform (GCP), click Google Config and update:
   a. **Project ID** to reflect the GCP project ID.
   b. **Default Deployment Tag** to reflect the environment name.
   c. **AuthJSON** to reflect the service account.

4. Click Create Networks and update the network names to reflect the network names for the new environment.
5. If your BOSH Director had an external hostname, you must change it in Director Config > Director Hostname to ensure it does not conflict with the hostname of the backed up Director.

6. Return to the Ops Manager Installation Dashboard and click the Pivotal Application Service (PAS) tile.

7. Click Resource Config. If necessary for your IaaS, enter the name of your new load balancers in the Load Balancers column.

8. If necessary, click Networking and update the load balancer SSL certificate and private key under Certificates and Private Keys for HAProxy and Router.

9. If your environment has a new DNS address, update the old environment DNS entries to point to the new load balancer addresses. For more information, see the Step 4: Configure Networking section of the Using Your Own Load Balancer topic and follow the link to the instructions for your IaaS.

10. If your PAS uses an external blobstore, ensure that the PAS tile is configured to use a different blobstore, otherwise it will attempt to connect to the blobstore that the backed up PAS is using.

11. Ensure your System Domain and Apps Domain under PAS Domains are updated to refer to the new environment’s domains.

12. Ensure that there are no outstanding warning messages in the BOSH Director tile, then disable Ops Manager advanced mode. For more information, see How to Enable Advanced Mode in the Ops Manager in the Pivotal Knowledge Base.

Step 4: Remove BOSH State File

1. SSH into your Ops Manager VM. For more information, see the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic.

2. To delete the /var/tempest/workspaces/default/deployments/bosh-state.json file, run the following on the Ops Manager VM:

   ```bash
   sudo rm /var/tempest/workspaces/default/deployments/bosh-state.json
   ```

3. In a browser, navigate to your Ops manager’s fully-qualified domain name.

4. Log in to Ops Manager.

Step 5: Deploy the BOSH Director

Use the Ops Manager API or the checkbox on the Review Pending Changes page to deploy the BOSH Director by itself.

Step 6: Transfer Artifacts to Jumpbox

In the Step 9: Back Up Your PAS Deployment section of the Backing Up Pivotal Cloud Foundry with BBR topic, in the After Taking the Backups section you moved the TAR and metadata files of the backup artifacts off your jumpbox to your preferred storage space. Now you must transfer those files back to your jumpbox.

Restore Your Backup

This section provides the steps you need to perform to restore your PCF backup with BBR.

Step 7: Retrieve BOSH Director Credentials

To use BBR, you must retrieve and record the following credentials:

- Bosh Director Credentials
- Bbr Ssh Credentials
- Uaa Bbr Client Credentials

There are two ways to retrieve BOSH Director credentials:
To retrieve your BOSH Director credentials using the Ops Manager Installation Dashboard, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the BOSH Director tile.
3. Click the Credentials tab.
4. Locate Director Credentials.
   a. Click Link to Credentials next to it.
   b. Verify the value of the `identity` field. It should be `director`.
   c. Copy and record the value of the `password` field.
5. Locate Bbr Ssh Credentials.
   a. Click Link to Credentials next to it.
   b. Copy the value of the `private_key_pem` field.
   c. Run the following command to reformat the copied value, and save it in the current directory to a file named `PRIVATE-KEY-FILE`:
   ```bash
   printf -- "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
   ```
Where:
- YOUR-PRIVATE-KEY is the text of your private key.
- PRIVATE-KEY-FILE is the path to the private key file you are creating.

For example:

```
$ printf "-----BEGIN RSA PRIVATE KEY-----MIIEkeycontents-----END RSA PRIVATE KEY-----" > bbr_key.pem
```

6. Locate the Uaa Bbr Client Credentials

a. Click Link to Credentials next to it.
b. Verify the value of the `identity` field. It should be `bbr_client`.
c. Record the value of the `password` field.

Option 2: Ops Manager API

To retrieve BOSH Director credentials using the Ops Manager API, perform the following steps:

1. Obtain your UAA access token. For more information, see Access the API.

2. Retrieve the Director Credentials by performing the following steps:

   a. Run the following command:

   ```
curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/director_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

   Where:
   - OPS-MAN-FQDN is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - UAA-ACCESS-TOKEN is your UAA access token.

   b. Verify the value of the `identity` field. It should be `director`.
   c. Record the value of the `password` field.

3. Retrieve the Bbr Ssh Credentials by performing the following steps:

   a. Run the following command:

   ```
curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
   -X GET \
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

   Where:
   - OPS-MAN-FQDN is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - UAA-ACCESS-TOKEN is your UAA access token.

   b. Copy the value of the `private_key_pem` field.
   c. Run the following command to reformat the copied value, and save it in the current directory to a file named `PRIVATE-KEY-FILE`:

   ```
   printf "YOUR-PRIVATE-KEY" > PRIVATE-KEY-FILE
   ```

   Where:
   - YOUR-PRIVATE-KEY is the text of your private key.
   - PRIVATE-KEY-FILE is the path to the private key file you are creating.

   For example:

   ```
   $ printf "-----BEGIN RSA PRIVATE KEY-----MIIEkeycontents-----END RSA PRIVATE KEY-----" > bbr_key.pem
   ```

4. Retrieve the Uaa Bbr Client Credentials by performing the following steps:

   a. Run the following command:
curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/uaa_bbr_client_credentials" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"

Where:
- OPS-MAN-FQDN is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
- UAA-ACCESS-TOKEN is your UAA access token.

b. Verify the value of the identity field. It should be bbr_client.

c. Record the value of the password field.

For more information about using the Ops Manager API, see the Using the Ops Manager API topic.

Step 8: Retrieve BOSH Director Address

Perform the following steps to retrieve the IP address of your BOSH Director from the BOSH Director tile:

1. If you are not using the Ops Manager VM as your jumpbox, install the latest BOSH CLI on your jumpbox.

2. From the Installation Dashboard in Ops Manager, select BOSH Director > Status and record the IP address listed for the Director. You access the BOSH Director using this IP address.

3. From the command line, log into the BOSH Director, using the IP address that you recorded above, by running the following command:

   ```
   bosh -e DIRECTOR-IP  
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE log-in
   ```

   Where:
   - DIRECTOR-IP is the BOSH Director IP address recorded above.
   - PATH-TO-BOSH-SERVER-CERTIFICATE is the path to the root Certificate Authority (CA) certificate as outlined in Step 1: Set Up your Jumpbox, above.

4. When prompted for Email (), specify director.

5. When prompted for Password (), enter the Director Credentials that you obtained in Retrieve BOSH Director Credentials.

   For example:
   ```
   $ bosh -e 10.0.0.3  
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate log-in
   Email (): director  
   Password (): **************  
   Successfully authenticated with UAA  
   Succeeded
   ```

Step 9: Restore the BOSH Director

Notes:
- The BBR BOSH Director restore command can take at least 15 minutes to complete.
1. SSH into your jumpbox. If you are using the Ops Manager VM as your jumpbox, see the Log in to the Ops Manager VM with SSH section of Advanced Troubleshooting with the BOSH CLI for procedures on how to use SSH to connect to the Ops Manager VM.

2. Ensure the BOSH Director backup artifact is in the folder you from which you will run BBR.

3. Run the BBR restore command from your jumpbox to restore the BOSH Director:

   bbr director
   --private-key-path PRIVATE-KEY-FILE
   --username bbr
   --host HOST
   restore
   --artifact-path PATH-TO-DIRECTOR-BACKUP

   Where:
   - PATH-TO-DIRECTOR-BACKUP is the path to the Director backup you want to restore.
   - PRIVATE-KEY-FILE is the path to the private key file you created in Step 7: Retrieve BOSH Director Credentials.
   - HOST is the address of the BOSH Director.
     - If the BOSH Director is public, HOST is a URL, such as https://my-bosh.xxx.cf-app.com.
     - If the BOSH Director is not public, HOST is the BOSH-DIRECTOR-IP retrieved in Step 8: Retrieve BOSH Director Address.
   - Use the optional --debug flag if you want to enable debug logs. See the Logging section of the Backing Up Pivotal Cloud Foundry with BBR topic for more information.

4. After the command succeeds, continue to Step 10: Identify Your Deployment.

   If the command fails, complete the following procedure:
   a. Run the BBR restore-cleanup command:

      bbr director
      --private-key-path PRIVATE-KEY-FILE
      --username bbr
      --host HOST
      restore-cleanup

      Where:
      - PRIVATE-KEY-FILE is the path to the private key file you created in Step 7: Retrieve BOSH Credentials.
      - HOST is the address of the BOSH Director.
        - If the BOSH Director is public, HOST is a URL, such as https://my-bosh.xxx.cf-app.com.
        - If the BOSH Director is not public, HOST is the BOSH-DIRECTOR-IP retrieved in Step 8: Retrieve BOSH Director Address.
   b. Run the BBR restore command again after checking the following:
      - All the parameters in the command are set.
      - The BOSH Director credentials are valid.
      - The specified deployment exists.
      - The source deployment is compatible with the target deployment.
      - That the jumpbox can reach the BOSH Director.

   Step 10: Identify Your Deployment

   1. Log in to your BOSH Director.

   2. To identify the name of the BOSH deployment that contains PCF, run the following command:

      bosh -e BOSH-DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE deployments

      Where:
      - BOSH-DIRECTOR-IP is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
For example:

```
$ bosh -e BOSH-DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE deployments
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Release()</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf-example</td>
<td>push-apps-manager-release/661.1.24</td>
</tr>
<tr>
<td></td>
<td>cf-backup-and-restore/0.0.1</td>
</tr>
<tr>
<td></td>
<td>binary-buildpack/1.0.11</td>
</tr>
<tr>
<td></td>
<td>capi/1.28.0</td>
</tr>
<tr>
<td></td>
<td>cf-autoscaling/91</td>
</tr>
<tr>
<td></td>
<td>cf-mysql/35</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

In the above example, the name of the BOSH deployment that contains PCF is `cf-example`. `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the root Certificate Authority (CA) certificate for the BOSH Director. If you are using the Ops Manager VM as your jumpbox, the path is `/var/tampest/workspaces/default/root_ca_certificate`.

**Step 11: Remove Stale Cloud IDs for All Deployments**

1. Review the deployments listed when performing Step 10: Identify Your Deployment.

2. To reconcile the BOSH Director's internal state with the state in the IaaS, run the following command for each deployment:

   ```
bosh -e BOSH-DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE -d DEPLOYMENT-NAME -n cck --resolution delete_disk_reference --resolution delete_vm_reference
   ```

   Where:
   - `BOSH-DIRECTOR-IP` is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine’s certificate chain.
   - `DEPLOYMENT-NAME` is the deployment name retrieved in Step 10: Identify Your Deployment.

3. To delete disk references, run the following command:

   ```
bosh cloud-check
   ```

   If the `bosh cloud-check` command does not successfully delete disk references, and you see a message similar to the following, perform the additional procedures in the Remove Unused Disks section below.

   Scanning 19 persistent disks: 19 OK, 0 missing ...

**Step 12: Redeploy PAS**

**Determine the Required Stemcells**

Perform either the following procedures to determine which stemcell is used by PAS:

- **Review the Stemcell Library:**
  1. Go to Stemcell Library.
  2. Record the PAS stemcell release number from the Staged column.

- **Review a Stemcell List Using BOSH CLI:**
  1. To retrieve the stemcell release using the BOSH CLI, run the following command:
bosh -e BOSH-DIRECTOR-IP deployments

Where BOSH-DIRECTOR-IP is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.

For example:

bosh -e BOSH-DIRECTOR-IP deployments
Using environment `10.0.0.5` as user `director` (bosh.*.read, openid, bosh.*.admin, bosh.read, bosh.admin)

<table>
<thead>
<tr>
<th>Name</th>
<th>Release(s)</th>
<th>Stencil(s)</th>
<th>Team(s)</th>
<th>Cloud Config</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf-9cb699587df4d7458</td>
<td>push-apps-manager-release/661.1.24</td>
<td>bosh-google-kvm-album-trusty-go_agent/3421.9</td>
<td></td>
<td>latest</td>
</tr>
</tbody>
</table>

For more information about stemcells in Ops Manager, see Importing and Managing Stemcells.

Upload Stemcells
1. Download the stemcell from Pivotal Network.
2. Run the following command to upload the stemcell used by PAS:

   ```
   bosh -e BOSH-DIRECTOR-IP -d DEPLOYMENT-NAME --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE upload-stemcell --fix PATH-TO-STEMCELL
   ```

   Where:
   - BOSH-DIRECTOR-IP is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
   - DEPLOYMENT-NAME is the deployment name retrieved in Step 10: Identify Your Deployment.
   - PATH-TO-BOSH-SERVER-CERTIFICATE is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine's certificate chain.
   - PATH-TO-STEMCELL is the path to your tile's stemcell.

   3. To ensure the stemcells for all of your other installed tiles have been uploaded, repeat the last step, running the `bosh upload-stemcell --fix PATH-TO-STEMCELL` command, for each stemcell that is different from the already uploaded PAS stemcell.

Redeploy PAS
1. From the Ops Manager Installation Dashboard, navigate to PAS Resource Config.
2. Ensure the number of instances for MySQL Server is set to 1.

   ![warning]
   Restore will fail if there is not exactly one MySQL Server instance deployed.

3. Ensure that all errands needed by your system are set to run.
4. Return to the Ops Manager Installation Dashboard.
5. Click Review Pending Changes.
6. Review your changes. For more information, see Reviewing Pending Product Changes.
7. Click Apply Changes to redeploy.

Step 13: (Optional) Restore Service Data

![warning]
BBR does not back up or restore any service data.

For this step, restore data to pre-provisioned service tiles.
The procedures for restoring service data vary. Consult the documentation for your service tile for more information.

For example, if you are using Redis for PCF v1.14, see Using BOSH Backup and Restore (BBR).

Step 14: Restore PAS

1. Refer to the table in Enabling External Blobstore Backups. If external blobstore support is not included in the version of Ops Manager and PAS you are using, restore the external blobstore with your IaaS-specific tools before restoring PAS.

2. Run the BBR restore command from your jumpbox to restore PAS:

   ```
   bbr deployment\n   --target BOSH-DIRECTOR-IP\n   --username BOSH-CLIENT\n   --password BOSH-PASSWORD\n   --deployment DEPLOYMENT-NAME\n   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE\n   restore\n   --artifact-path PATH-TO-PAS-BACKUP
   ```

   Where:
   - `BOSH-DIRECTOR-IP` is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
   - `BOSH-CLIENT`, `BOSH-PASSWORD` are the Uaa Bbr Client Credentials, identity and password, that you retrieved in Step 7: Retrieve BOSH Director Credentials.
   - `DEPLOYMENT-NAME` is the deployment name retrieved in Step 10: Identify Your Deployment.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE` is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine’s certificate chain.
   - `PATH-TO-PAS-BACKUP` is the path to the PAS backup you want to restore.

3. If desired, scale the MySQL Server job back up to its previous number of instances by navigating to the Resource Config section of the PAS tile. After scaling the job, return to the Ops Manager Installation Dashboard.

4. Click Review Pending Changes.

5. Review your changes. For more information, see Reviewing Pending Product Changes.

6. Click Apply Changes to deploy.

Step 15: (Optional) Restore On-Demand Service Instances

If you have on-demand service instances provisioned by an on-demand service broker, perform the following steps to restore them after successfully restoring PAS:

1. Navigate to an on-demand service tile in the Installation Dashboard.

2. Select the Errands tab.

3. Ensure the Upgrade All Service Instances errand is On.

4. Repeat for all on-demand service tiles.

5. Return to the Installation Dashboard.

6. Click Review Pending Changes, review your changes, and then click Apply Changes. For more information, see Reviewing Pending Product Changes. This will include running the Upgrade All Service Instances errand for the on-demand service, which will redeploy the on-demand service instances.

7. (Optional) Restore service data to every on-demand service instance.

Notes:
- The BBR PAS restore command can take at least 15 minutes to complete.
- Pivotal recommends that you run it independently of the SSH session, so that the process can continue running even if your connection to the jumpbox fails. The command above uses `nohup`, but you could also run the command in a `screen` or `tmux` session.
8. Any app on PAS bound to an on-demand service instance may need to be restarted to start consuming the restored on-demand service instances.

Step 16: (Optional) Restore Non-Tile Deployments

If you have any deployments that were deployed manually with the BOSH Director rather than through an Ops Manager tile, perform the following steps to restore the VMs.

1. To obtain a list of all deployments on your BOSH Director, run the following command:

   ```
   bosh -e BOSH-DIRECTOR-IP\n   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE\n   deployments
   ```

   Where:
   - **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
   - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine's certificate chain.

2. Identify the names of the deployments that you need to restore. Do not include the deployments from Ops Manager tiles.

3. Run the following command for each deployment you need to restore:

   ```
   bosh -e BOSH-DIRECTOR-IP\n   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE\n   -d DEPLOYMENT-NAME\n   cck --resolution=recreate_vm
   ```

   Where:
   - **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
   - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine's certificate chain.
   - **DEPLOYMENT-NAME** is the deployment name retrieved in Step 10: Identify Your Deployment.

4. Run the following command to verify the status of the VMs in each deployment:

   ```
   bosh -e BOSH-DIRECTOR-IP\n   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE\n   -d DEPLOYMENT-NAME\n   vms
   ```

   Where:
   - **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
   - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine's certificate chain.
   - **DEPLOYMENT-NAME** is the deployment name retrieved in Step 10: Identify Your Deployment.

The process state for all VMs should show as running.

After Restoring Your Backup

This section provides the steps you need to perform after restoring your PCF backup with BBR.

Step 17: Remove Unused Disks

⚠️ **warning:** This is a very destructive operation.

Disks from a previous deployment will prevent recreated deployments from working.
Use BOSH To Clean Up Disks

1. To clean up disk references, run the following command:

   ```
   bosh cloud-check
   ```

Manually Clean Up Disks

If `bosh cloud-check` does not clean up all disk references, you must manually delete the remaining disks.

1. To delete the remaining disks, perform one of the following procedures:

   1. Use the BOSH CLI to delete the disks by performing the following steps:

      1. Target the redeployed BOSH Director using the BOSH CLI by performing the procedures in Step 8: Retrieve BOSH Director Address.
      2. List the deployments by running the following command:

         ```
         bosh -e BOSH-DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE deployments
         ```

         Where:

         - **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
         - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine’s certificate chain.

      3. Delete each deployment with the following command:

         ```
         bosh -d DEPLOYMENT-NAME delete-deployment
         ```

         Where:

         - **DEPLOYMENT-NAME** is the deployment name retrieved in Step 10: Identify Your Deployment.

   2. Log in to your IaaS account and delete the disks manually.

   1. To retrieve a list of disk IDs, run the following command:

      ```
      bosh -e BOSH-DIRECTOR-IP --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE instances -i
      ```

      Where:

      - **BOSH-DIRECTOR-IP** is the BOSH Director IP retrieved in Step 8: Retrieve BOSH Director Address.
      - **PATH-TO-BOSH-SERVER-CERTIFICATE** is the path to the Certificate Authority (CA) certificate for the BOSH Director, if the certificate is not verifiable by the local machine’s certificate chain.

Once the disks are deleted, continue with Step 11: Remove Stale Cloud IDs for All Deployments.
Enabling External Blobstore Backups

This topic provides instructions for enabling external blobstore backups in your Pivotal Application Service (PAS) tile.

BOSH Backup and Restore (BBR) supports the following:

- Versioned S3 or S3-Compatible Blobstores
- Unversioned S3 or S3-Compatible Blobstores
- Azure Blobstores

For more information, see Backup and Restore with External Blobstores

Note: To enable external blobstore backups for PAS, the Backup Restore Node must be enabled. See Enable Backup Restore Node in Backing Up Pivotal Cloud Foundry with BBR.

External Blobstore Support

External blobstore backup support varies based on which version of Ops Manager you are running and what type of blobstore you are backing up.

In some cases, external blobstore support is included in the version of Ops Manager you are using. In other cases, installing Blobstore Add-On is required.

Refer to the table below to determine if external blobstore support is included in the version of Ops Manager you are using.

<table>
<thead>
<tr>
<th>Ops Manager Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Versioned</td>
</tr>
<tr>
<td>Unversioned</td>
</tr>
<tr>
<td>Azure</td>
</tr>
</tbody>
</table>

S3 or S3-compatible Blobstores

In PAS v2.2 and later, BBR supports backing up and restoring versioned and unversioned S3 or S3-compatible blobstores by default. For more information about configuring an S3 or S3-compatible blobstore, see Configuring Load Balancing for PAS.

Note:
- Backup artifacts of external, S3-compatible, versioned blobstores do not contain the physical blobs. BBR requires that the original buckets still exist to be restored.
- To protect yourself from losing a bucket, see Enable Replication on Your External Blobstore in Backup and Restore with External Blobstores in the Cloud Foundry documentation.

Azure Blobstores

To configure your Azure blobstore for backups, you must enable Soft Delete in your Azure Storage account. You can also enable Soft Delete for existing Azure storage accounts that were not initially created with Soft Delete enabled. For more information, see Soft delete for Azure Storage blobs in the Microsoft documentation.

After enabling Soft Delete, you can enable Azure blobstore backups in Ops Manager for new or existing blobstores. For more information, Configuring Load Balancing for PAS.

To save storage space and cost, Pivotal recommends that you configure a retention policy to permanently delete objects after a period of time.
Troubleshooting BBR

This topic lists common troubleshooting scenarios and their solutions when using BOSH Backup and Restore (BBR) to back up and restore Pivotal Cloud Foundry (PCF).

Troubleshooting During a Restore

This section provides solutions for errors that occur during a restore.

Restore Fails with a MySQL Monit Start Timeout

**Symptom**

While running the BBR restore command, restoring the job $mysql-restore$ fails with the following error:

```
1 error occurred:
  * restore script for job mysql-restore failed on mysql/0.

Monit start failed: Timed out waiting for monit: 2m0s
```

**Explanation**

This happens when the MySQL job fails to start within the timeout period. It ends up in an **Execution Failed** state and **monit** never tries to start it again.

**Solution**

Ensure that your MySQL Server cluster has only one instance. If there is more than one instances of MySQL Server, the restore fails with a **monit** start timeout. Scale down to one instance and retry.

If your MySQL Server cluster is already scaled down to one node, it may have taken longer than normal to restart the cluster. Follow the procedure below to manually verify and retry.

1. To list the VMs in your deployment, run the following command:

   ```
   bosh -e BOSH-DIRECTOR-IP --ca-cert /var/tempest/workspaces/default/root_ca_certificate \
   -d DEPLOYMENT-NAME \
   ssh
   ```

   Where:
   - **BOSH-DIRECTOR-IP** is the BOSH Director IP. For how to retrieve this IP address, see Retrieve BOSH Director Address.
   - **DEPLOYMENT-NAME** is the deployment name. For how to retrieve the deployment name, see Identify Your Deployment.

2. Locate the **mysql** VM.

3. SSH into the **mysql** VM.

4. To check that the MySQL job process is running, perform one of the following from the **mysql** VM:

   - If you selected **Internal Databases - MySQL - Percona XtraDB Cluster** when you configured the PAS tile, run the following command:
     ```
     ps aux | grep galera-init
     ```

   - If you selected **Internal Databases - MySQL - MariaDB Galera Cluster** when you configured the PAS tile, run the following command:
5. Run the following command to check that `monit` reports that the MySQL job process is in an **Execution Failed** state:

```
sudo monit summary
```

6. If so, run the following command from the `mysql` VM to disable monitoring:

```
monit unmonitor
```

7. Run the following command to enable monitoring:

```
monit monitor
```

8. After a few minutes, run the following command:

```
monit summary
```

The command should report that all the processes are running.

9. Re-attempt the restore with BBR.

---

### Deployment Does Not Match the Structure of the Backup

**Symptom**

The following error displays:

```
Deployment 'deployment-name' does not match the structure of the provided backup
```

**Explanation**

The instance groups with the restore scripts in the destination environment don’t match the backup metadata. For example, they may have the wrong number of instances of a particular instance group, or the metadata names an instance group that does not exist in the destination environment.

**Solution**

BBR only supports restoring to an environment that matches your original environment. Pivotal recommends altering the destination environment to match the structure of the backup.

---

### General Troubleshooting

This section provides solutions for general errors.

**Connection Error**
Symptom
BBR displays an error message containing “SSH Dial Error” or another connection error.

Explanation
The jumpbox and the VMs in the deployment are experiencing connection problems.

Solution
Perform the following steps:

1. To ensure your deployment is healthy, run the following command:
   ```
   bosh vms
   ```

2. To clean up the data from the failed backup on the instances, run the following command:
   ```
   bbr deployment backup-cleanup
   ```

   Note: This step must be performed, otherwise, further BBR commands fail.

3. Repeat the BBR operation.

Error Running Metadata Script

Symptom
BBR backup or restore fails with a metadata error:

```
1 error occurred:
  error 1:
  An error occurred while running metadata script for job redis-server on redis/8ce9b31f-1756-480b-8e3e-a4609b14b6a6: error from metadata
```

Explanation
There is a problem with your PCF install.

Solution

Contact Pivotal Support
Using Ops Manager

Ops Manager is a web application that you use to deploy and manage a Pivotal Cloud Foundry (PCF) PaaS. This is a guide to deploying and using Ops Manager.

Ops Manager UI

Use the Ops Manager UI to configure settings for your Ops Manager Installation.

For more information on the Ops Manager UI, see Using the Ops Manager UI.

Ops Manager API

Use the Ops Manager API to configure any Ops Manager task.

See the Using the Ops Manager API topic to learn how to get started using the Ops Manager API.

Related Links

Using the Ops Manager UI and Installed Products

- Using the Ops Manager Interface
- Adding and Deleting Products
- Retrieving Credentials from Your Deployment
- Floating Stemcells
- Configuring BOSH Director on vSphere
- vSphere Service Account Requirements
- Creating UAA Clients for BOSH Director
- Configuring BOSH Director on OpenStack
- Using Your Own Load Balancer
- Pivotal Cloud Foundry User Types
- Creating and Managing Ops Manager User Accounts
- Creating New PAS User Accounts
- Logging In to Apps Manager
- Adding Existing SAML or LDAP Users to a Pivotal Cloud Foundry Deployment
- Deleting an AWS Installation from the Console
- Modifying Your Ops Manager Installation and Product Template Files
- Managing Errands in Ops Manager

Using the Ops Manager API

- Configuring the BOSH Director with the Ops Manager API
- Configuring a Product with the Ops Manager API

Backing Up

- Backing Up and Restoring Pivotal Cloud Foundry
Monitoring, Logging, and Troubleshooting

- Monitoring Virtual Machines in Pivotal Cloud Foundry
- Diagnosing Problems in PCF
- Troubleshooting Problems in PCF
- Troubleshooting Ops Manager for VMware vSphere
- Recovering MySQL from PAS Downtime
- Advanced Troubleshooting with the BOSH CLI
Using the Ops Manager API

This topic explains how to start using the Ops Manager API.

Platform operators use the Ops Manager API to automate deployments, retrieve and manage credentials, and otherwise work under the hood of the Ops Manager interface.

Tile developers use the Ops Manager API to test and debug Pivotal Cloud Foundry (PCF) product tiles.

For the complete Ops Manager API documentation, see either of the following:

- [https://docs.pivotal.io/pivotalcf/2-5/opsman-api](https://docs.pivotal.io/pivotalcf/2-5/opsman-api)
- [https://YOUR-OPS-MANAGER-FQDN/docs](https://YOUR-OPS-MANAGER-FQDN/docs), adding /docs to the URL of your Ops Manager

Requirements

You must install the User Account and Authentication Command Line Interface (UAAC) to perform the procedures in this topic. To install the UAAC, run the following command from a terminal window:

```
$ gem install cf-aac
```

Step 1: Authenticate

To use the Ops Manager API, you must authenticate and retrieve a token from the Ops Manager User Account and Authentication (UAA) server. For more information about UAA, see the [User Account and Authentication (UAA) Server](#) topic.

Perform the procedures in the [Internal Authentication](#) or [External Identity Provider](#) section below depending on which authentication system you configured for Ops Manager.

Internal Authentication

If you configured your Ops Manager for Internal Authentication, perform the following procedures specific to your IaaS:

vSphere

You need the credentials used to import the PCF .ova or .ovf file into your virtualization system.

1. From a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.

2. When prompted, enter the password that you set during the .ova deployment into vCenter. For example:

```
$ ssh ubuntu@my-opsmanager-fqdn.example.com
Password: ***********
```

3. Proceed to Authenticate into Ops Manager.

AWS, Azure, and OpenStack

1. Locate the Ops Manager FQDN on the AWS EC2 instances page, Azure Virtual machines page, or the OpenStack Access & Security page.

2. Run `chmod 600 ops mgr.pem` to change the permissions on the .pem file to be more restrictive:

```
$ chmod 600 ops mgr.pem
```

3. Run `ssh -i ops mgr.pem ubuntu@OPS-MANAGER-FQDN` to SSH into the Ops Manager VM. Replace `OPS-MANAGER-FQDN` with the fully qualified domain
name of Ops Manager. For example:

```
$ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
```

4. Proceed to **Authenticate into Ops Manager**.

**GCP**

1. Confirm that you have installed the gcloud CLI. If you do not have the gcloud CLI, see the Google Cloud Platform documentation.

2. Run `gcloud config set project MY-PROJECT` to configure your Google Cloud Platform project. For example:

```
$ gcloud config set project gcp
```

3. Run `gcloud auth login MY-GCP-ACCOUNT`. For example:

```
$ gcloud auth login user@example.com
```

4. Run `gcloud compute ssh MY-INSTANCE --zone MY-ZONE`. For example:

```
$ gcloud compute ssh om-pcf-1a --zone us-central1-b
```

5. Run `sudo su - ubuntu` to switch to the `ubuntu` user.

6. Proceed to **Authenticate into Ops Manager**.

**Authenticate Into Ops Manager**

1. After successfully SSHing into the Ops Manager VM, use the UAAC to target your Ops Manager UAA server:

```
$ uaac target https://OPS-MAN-FQDN/uaa
```

2. Retrieve your token to authenticate:

```
$ uaac token owner get
Client ID: opsman
Client secret: [Leave Blank]
User name: OPS-MAN-USERNAME
Password: OPS-MAN-PASSWORD
```

Replace **OPS-MAN-USERNAME** and **OPS-MAN-PASSWORD** with the credentials that you use to log in to the Ops Manager web interface.

**External Identity Provider**

If you configured your Ops Manager for an external Identity Provider with SAML or LDAP, do the following:

1. From your local machine, target your Ops Manager UAA server:

```
$ uaac target https://OPS-MAN-FQDN/uaa
```

2. Retrieve your token to authenticate. When prompted for a passcode, retrieve it from `https://OPS-MAN-FQDN/uaa/passcode`.

```
$ uaac token sso get
Client ID: opsman
Client secret: [Leave Blank]
Passcode: YOUR-PASSCODE
```

If authentication is successful, the UAAC displays the following message: **Successfully fetched token via owner password grant.**
Step 2: Access the API

Ops Manager uses authorization tokens to allow access to the API. You must pass an access token to the API endpoint in a header that follows the format

```
Authorization: Bearer YOUR-ACCESS-TOKEN
```

The following example procedure retrieves a list of deployed products. See the Ops Manager API documentation for the full range of API endpoints.

If you use Internal Authentication, you must perform the following procedures from the Ops Manager VM. If you use an External Identity Provider, you may perform the procedures from your local machine.

1. List your tokens:

   $ uaa contexts

   Locate the entry for your Ops Manager FQDN. Under client_id: opsman, record the value for access_token.

2. Use the `GET /api/v0/deployed/products` endpoint to retrieve a list of deployed products, replacing `UAA-ACCESS-TOKEN` with the access token recorded in the previous step:

   ```
   ```

   The request produces the following response:

   ```
   ["installation_name":"p-bosh","guid":"p-bosh-00000000000000000000","type":"p-bosh","product_version":"1.10.0.0"],"installation_name":"cf-00000000000000000000","guid":"cf-00000000000000000000","type":"cf","product_version":"1.10.0"
   ```
Configuring the BOSH Director with the Ops Manager API

Overview

This topic describes how to configure the BOSH Director using the Ops Manager API. Advanced PCF operators and administrators can do a complete end-to-end setup of Ops Manager using the API. This procedure is not intended for beginner users of Ops Manager, PCF, or API-based procedures.

For information about setting up Ops Manager with the UI, see Using Ops Manager.

Prerequisites

Before you configure the BOSH Director with the Ops Manager API, you must have the following components downloaded or accessible on your local host:

- A current version of Ops Manager downloaded from Pivotal Network.
- Access to the Ops Manager API. For more information about the Ops Manager API, see Using the Ops Manager API.
- An external SAML or LDAP client.
- An IaaS provider configured for use with Ops Manager. For more information about configuring an IaaS, see Planning Your PCF Deployment.

Setting Up the BOSH Director

To configure the BOSH Director with the Ops Manager API, do the following:

1. Access the Ops Manager API by following the procedure in Using the Ops Manager API.

2. Enter the following HTTP request to perform initial setup: `POST /api/v0/setup`.
   Include metadata in your request to specify configuration details for your deployment. For example, specify the deployment's decryption passphrase, EULA status, and proxy information, if applicable.
   For more information about configuring this request, see Setting up with an internal usersotre.

3. A 200 OK response appears.

   After the successful response, a UAA client with the metadata you specified is created and launched. Ops Manager users can authenticate with UAAC, and if they have a pre-created client they can target UAAC with the client name and password.
   For more information on pre-created clients, see Creating and Managing Ops Manager User and Client Accounts.

4. Enter the following HTTP request to configure your IaaS and BOSH Director: `PUT api/v0/staged/director/properties`.

   There are many configuration parameters available to customize your BOSH Director. All the commands are IaaS-agnostic except the IaaS configuration key. Missing required fields should cause an error.
   For more information about configuring the BOSH Director, see Fetching Director, IaaS, and Security Properties.

5. Enter the following HTTP request to create Availability Zones (AZs) for your product: `POST api/v0/staged/director/availability_zones`.

   For more information about creating AZs, see Creating availability zones.

6. Enter the following HTTP request to create networking rules for the deployment: `PUT api/v0/staged/director/networks`.

   Specify whether or not to use ICMP checks by setting the `icmp_checks_enabled` parameter to true or false.

Note: You do not need to specify UAA or other authentication details on a first-time deploy. Making the setup request automatically creates a UAA client when it completes successfully. If you specify a UAA client for a first-time deployment, the deployment will fail.

Note: The create-azs endpoint is optional for Azure-based deployments, because you cannot manually configure Azure AZs.
For more information about networking rules, see Updating networks (Experimental).

7. Enter the following HTTP request to assign a singleton AZ and a network where your BOSH Director will be located:

```
PUT api/v0/staged/director/network_and_az
```

For more information about configuring AZs and networks, see Updating network and availability zone assignments.

8. You must update the BOSH Director’s resource configuration settings before deploying Ops Manager. Resource configuration is highly variable depending on the requirements of your deployment.

For more information about updating the resource configuration, see Configuring resources for a job (Experimental).

To update the resource configuration, do the following:

a. Use the following HTTP request to find your Director’s GUID:

```
GET api/v0/staged/products
```

b. Use the following HTTP request to list all jobs on the Director:

```
GET api/v0/staged/products/:BOSH-DIRECTOR-GUID/jobs
```

c. Use the following HTTP request to update the resource config of each job on the BOSH Director:

```
PUT api/v0/staged/products/:BOSH-DIRECTOR-GUID/jobs/:JOB-GUID/resource_config
```

9. BOSH Director configuration is now complete. You can deploy Ops Manager now, or continue with further configuration.

Use the following HTTP request to “Apply Changes” and deploy the BOSH Director:

```
POST /api/v0/installations
```

Next Steps

After you have configured the BOSH Director, you can continue configuring Ops Manager with other API endpoints.

- Configuring a Product with the Ops Manager API
- Uploading and Assigning a Stemcell with the Ops Manager API
- Upgrading Ops Manager with the Ops Manager API
Configuring a Product with the Ops Manager API

Overview

This topic describes how to configure product tiles for Ops Manager using the Ops Manager API. This procedure is not intended for beginner users of Ops Manager, PCF, or API-based procedures.

For information about setting up Ops Manager with the UI, see Using Ops Manager.

Prerequisites

Before you configure a product tile with the Ops Manager API, you must have completed the following prerequisites:

- A BOSH Director configured and staged for deployment. To learn about configuring the BOSH Director with the API, see Configuring the BOSH Director with the Ops Manager API.
- A .pivotal file for the product you wish to configure. For more information about creating a .pivotal file, see PCF Tile Developer Guide.

Configuring a Product with the API

1. Upload or import the product into Ops Manager.
   a. Enter the following HTTP request to upload the product: POST api/v0/available_products
   b. Enter the following HTTP request to list the names of available products: GET api/v0/available_products
   c. Find the product you uploaded by referencing its name and version from the list that displays.
   d. Enter the following HTTP request to add the uploaded product to Ops Manager: POST /api/v0/staged/products

2. Confirm that the product is staged for deployment. Enter the following HTTP request to confirm the product exists in Ops Manager and display the product GUID: GET api/v0/staged_products

3. Enter the following HTTP request to assign availability zones (AZs) and networks to the product:
   
   PUT /api/v0/staged/products/:PRODUCT-GUID/networks_and_azs

4. Enter the following HTTP request to update the product's properties:
   
   PUT /api/v0/staged/products/:PRODUCT-GUID/properties

5. Enter the following HTTP request to configure syslog for the product:
   
   PUT /api/v0/staged/products/:PRODUCT-GUID/syslog_configuration

6. Update the product's resource configuration:

   a. Enter the following HTTP request to list all jobs on a product and display the GUID for each job:
      
      GET api/v0/staged/products/:PRODUCT-GUID/jobs
   
   b. Enter the following HTTP request to update the resource config for a particular job:
      
      PUT api/v0/staged/products/:PRODUCT-GUID/jobs/:JOB-GUID/resource_config

QUESTIONS:

- Can we get code samples with parameters for some of these? 'Modify the resource config with this endpoint' feels really sparse–I want some details about what exactly they can do, or at least links to API docs with code samples for same.
Using the Ops Manager UI

Overview

This topic explains how to start using the Ops Manager UI, and lists other content about the Ops Manager UI.

Ops Manager operators can configure Ops Manager using a dashboard-style interface. The Ops Manager UI lets you make installation-wide modifications to Ops Manager, including updates to authentication methods, security and networking configuration, and more.

Advanced users can configure Ops Manager by accessing the Ops Manager API. For information about configuring Ops Manager with the API, see Using the Ops Manager API.

Access the Ops Manager UI

Ops Manager is accessible at its unique URL, which was assigned during the initial PCF setup process. For more information about setting up PCF, see Architecture and Installation Overview.

Browser Support

The Ops Manager UI is accessible from any internet-enabled host. It is compatible with current and recent versions of all major browsers.

Pivotal recommends using the current version of Chrome, Firefox, or Safari for the best Ops Manager experience.

Related Links

- Using the Ops Manager Interface
- Reviewing Pending Product Changes
- Adding and Deleting Products
- Importing and Managing Stemcells
- Logging in to Apps Manager
- Managing Errands in Ops Manager

© Copyright Pivotal Software Inc, 2013-2019

659
This topic describes key features of the Pivotal Cloud Foundry (PCF) Operations Manager interface.

Installation Dashboard Page

The Installation Dashboard is the central Ops Manager interface for managing your PCF deployment. It displays the BOSH Director configuration for your IaaS and product tiles for your PCF deployment.

The following screenshot shows the Installation Dashboard. Each section is labeled with a red letter. Click the image to see it at full size.

The following list describes each labeled section of the Installation Dashboard:

- **A — Import a Product**: Click this button to add a new product to Ops Manager. You can download PCF-compatible product files from Pivotal Network, *if you have configured a Pivotal Network API token in the Settings section of Ops Manager, a list of the latest versions of already imported products appears automatically.*

- **B — Delete All Unused Products**: Click this link to delete any unused products.

- **C — Installation Dashboard**: Click this link to return to the Installation Dashboard page from other Ops Manager pages.

- **D — Stemcell Library**: Click this link to open the Stemcell Library. In the Stemcell Library you can import stemcells, stage stemcells, and review your stemcell version numbers. For more information, see Importing and Managing Stemcells.

- **E — Change Log**: Click this link to view and search a log of your previous installations. See the Change Log Page section of this topic for details.

- **F — User Account Menu**: Click the arrow next to your username to view the menu. Use this menu to navigate to your Settings page, view My Account to change your email and password, or log out of the Installation Dashboard. See My Account of this topic for details on account information. See the Settings Page section of this topic for details on available Ops Manager settings.

- **G — Revert**: Click the revert button to return to the previous successful installation. Any pending changes you have made to the current installation settings will be lost.

- **H — Review Pending Changes**: Click the button to go to the Review Pending Changes page, which organizes pending changes by tile. You are able to enable or disable each tile to selectively deploy individual tiles and their dependencies. For more information, see Reviewing Pending Product Changes.

- **I — Orange Bar**: Indicates that additional configuration for the product tile is required before deployment. Click the product tile to complete its configuration.
J — Missing Stemcell Link: If an imported product is missing a required stemcell, a Missing Stemcell link appears on the tile. Click this link to open the Stemcell Library. For more information, see Importing and Managing Stemcells.

K — API Docs: Click this link to go to the Ops Manager API documentation, which details how you can manage Ops Manager through the API rather than with the user interface. For more information about the Ops Manager API, see Using the Ops Manager API.

Change Log Page

Navigate to the Change Log page by clicking the corresponding link in the dashboard header. This page lets you view changes between current and past deployments.

The table columns display attributes associated with each deployment:

- **Started**—The date and time, in UTC format, when the deployment began.
- **Finished**—The date and time, in UTC format, when the deployment ended.
- **User**—The user that initiated the deployment.
- **Status**—The status of this deployment.
- **Added**—The tiles that were newly added to the build.
- **Updated**—The tiles that were changed from the previous build.
- **Deleted**—The tiles that were removed from the previous build.
- **Unchanged**—The tiles that were not changed between deployments.
- **Logs**—A link to the Installation Log for the respective entry.

Using the Change Log Page

Configure the Change Log page by modifying the following fields:

- **Installation Dashboard**—Click Installation Dashboard to return to Ops Manager's Installation Dashboard. Alternatively, click the Back button in your web browser.
- **Show X entries**—Click the number displayed in the Show X entries dropdown to choose between 10, 25, 50, and 100 entries.
- **Search**—Type in the search box to sort the Change Log page by text or integer matches. As you type, matching entries appear on the screen.
- **Previous/Next**—Click Previous, Next, or the number between them to load older or newer entries.

Settings Page

Navigate to the Settings page by clicking on your user name located at the upper right corner of the screen and selecting Settings.

The Settings configuration screen displays.
The following section describes each configuration pane:

**Note:** Modifying these settings does not require you to return to the Installation Dashboard and click **Apply Changes**. These settings apply to the Ops Manager VM. The BOSH Director does not apply them to your PCF deployment.

### Change Decryption Passphrase

Use this pane to reset your decryption passphrase.

### SAML Settings

To change your Identity Provider (IdP) to SAML, configure the following fields:

- **Current Decryption Passphrase:** Enter your decryption passphrase.
- **SAML IDP Metadata:** Enter the full URL or XML SAML IdP metadata.
- **BOSH IDP Metadata:** (Optional) Enter the full URL or XML BOSH IdP metadata. If left blank, the default is the same metadata as the field above.
- **SAML Admin Group:** Enter the name of the SAML group that contains all of the Ops Manager administrators. This field is case-sensitive.
- **Groups Attribute:** Enter the groups attribute tag name with which you configured the SAML server. This field is case-sensitive.
- **Provision an admin client in the Bosh UAA:** Enable if you want to provision an admin client in the BOSH UAA. For more information, see [Provision Admin Client](#) in the Creating UAA Clients for BOSH Director topic.

### LDAP Settings

Use this pane to change your IdP to LDAP.

For more information about changing your IdP to SAML or LDAP, view the following instructions for your IaaS configuration:

- Amazon Web Services. See [Configuring BOSH Director on AWS](#).
- Google Cloud Platform. See [Configuring BOSH Director on GCP Manually](#).
- Microsoft Azure. See [Configuring BOSH Director on Azure Manually](#).
- OpenStack. See [Configuring BOSH Director on OpenStack](#).
- vSphere. See [Configuring BOSH Director on vSphere](#).
SSL Certificate

Configure Ops Manager to use a custom SSL certificate for all Ops Manager traffic both through the UI and API.

If you leave the fields blank, Ops Manager uses an auto-generated self-signed certificate rather than your own custom certificate and private key. If you wish to replace a custom certificate with a self-signed certificate, select Revert to self-signed certificate.
**Pivotal Network Settings**

Enter your Pivotal Network API token and click Add Token to connect your Installation Dashboard to Pivotal Network.

**Proxy Settings**

If you are using a proxy to connect to Ops Manager, update your Proxy Settings by providing a HTTP proxy, HTTPS proxy, or No proxy.
Custom Banner

Create a custom text banner to communicate important messages to operators. For **UI Banner**, enter the text you want to be shown on each page of the Ops Manager UI. For **SSH Banner**, enter the text that appears when an operator shells into Operations Manager.

Export Installation Settings

Exports the current installation with all of its assets. When you export an installation, the exported file contains references to the base VM images, necessary packages, and configuration settings.

Syslog

Viewable by administrators only. Configure a custom Syslog server for Ops Manager. When you select Yes and fill the following fields, Ops Manager produces and sends all Ops Manager logs to the configured syslog endpoint.

1. Select **Syslog**.
Syslog

Do you want to configure Syslog for Bosh Director?

- No
- Yes

Address*

The address or host for the syslog server

Port*

Transport Protocol*

TCP

Enable TLS

Permitted Peer*

SSL Certificate*

Queue Size

100000

Save

2. (Optional) Select Yes to send BOSH Director system logs to a remote server.
3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**

5. Select **TCP** or **UDP** from the **Transport Protocol** dropdown. This selection determines which transport protocol is used to send the logs to the remote server.

6. (Optional) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.

   **Note:** Pivotal strongly recommends that you enable TLS encryption when you are forwarding logs. Logs can contain sensitive information, such as cloud provider credentials.

7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.

8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.

9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** field. This field requires the rainerscript syntax.

10. Click **Save Syslog Settings**.

### Advanced Options

- **Download Activity Data**: Downloads a directory containing the config file for the installation, the deployment history, and version information.
- **Download Root CA Cert**: Use this to download the root CA certificate of your deployment as an alternative to curling the Ops Manager API.
- **View Diagnostic Report**: Displays various types of information about the configuration of your deployment.
- **Delete This Installation**: Deletes your Ops Manager installation permanently.

### My Account Page

To change your email and password, navigate to the **My Account** page by clicking on your user name located at the upper right corner of the screen and selecting **My Account**.
Reviewing Pending Product Changes

This topic describes the Review Pending Changes page and how to selectively deploy individual or multiple tiles in Ops Manager. Deploying tiles selectively can drastically reduce deployment time.

Review Pending Changes Page

The following screenshot shows the Review Pending Changes page. Features called out with red letters are described below.

Each labeled section of the Review Pending Changes page works as follows:

- **A — Select All Products:** When enabled, the checkbox selects all available products. When disabled, the checkbox deselects all available products.
- **B — Select Product:** The checkboxes next to each product listing allow you to select the tiles you want to deploy. The BOSH Director is always selected.
- **C — Product Listing:** Each product listing displays the version of the product staged for deployment. The top of each listing shows one of the following three colors:
  - Green—Tile is configured and ready for deployment.
  - Orange—Tile is not fully configured and cannot be deployed yet.
  - Red—Tile is pending deletion.
- **D — Depends on:** This section lists tile dependencies and versions. If you choose to deploy a tile that has dependencies, you must also deploy all of that tile’s dependencies in the same deployment.
**E—Warnings:** Red text in orange product listings indicates changes you need to make before you deploy the tile. You might see some of the following warnings:

- **Missing stemcell**—You must import and stage a stemcell with Stemcell Library. The warning links to the Stemcell Library.
- **Stemcell out of date**—You must update the stemcell of your tile in Stemcell Library. For more information, see Importing and Managing Stemcells.
- **Configuration is incomplete**—You must complete the configuration of your tile.
- **Configuration is invalid**—You must resolve your tile’s invalid configurations.

**F—Changes:** This section lists high-level pending changes to the product that Ops Manager has not yet attempted to deploy. When Ops Manager tries to deploy a product but the deployment fails, it clears this list and the pending changes are retained for the next deployment. To see if your product deployed successfully, see Change Log Page.

Listed changes include:

- Tile deletion is staged.
- Tile version is updated.
- Stemcell is updated.

**G—Errands:** If you expand this section, you can enable or disable errands that trigger when that tile deploys. For example, you may enable an errand to log properties. For more information, see Managing Errands in Ops Manager.

**H—Staged:** This section indicates the status of the tile.

**I—See Changes:** Clicking this button takes you to the Pending Changes page for that tile.

**J—Apply Changes:** Clicking this button deploys all enabled tiles.

---

**About Pending Changes for a Specific Tile**

If you have changed the configuration of a tile in your deployment, use the Pending Changes page to review the changes to the existing manifest or configs line-by-line.

This page summarizes staged changes to the existing tile’s configuration. The changes displayed in this page have not yet been applied to the deployment.

This page allows operators to verify changes to the manifest and configs so they can approve any staged changes before applying changes to their deployment.

**Use the Pending Changes Page**

The change summary fields display features that have been enabled or disabled using the Ops Manager UI or API. These changes are staged, but have not been implemented yet. You must click Apply Changes to put these changes into effect.

If you want to see changes for a specific tile, do the following:


2. Review the changes displayed. Settings that have been disabled are displayed in red with a minus sign (-). Settings that have been enabled are displayed in green with a plus sign (+).

3. Close the tab and return to the Review Pending Changes page to apply the changes or review other tiles.

The screenshot below shows changes to the BOSH Director’s manifest.
When you make changes to a tile, change summaries display for the following tile areas:

- Manifest
- Runtime configs
- CPI configs
- Cloud configs

If an aspect of the tile has not changed, the change summary displays `No changes`.

**Note:** Changes to errands do not display in the page UI.
Adding and Deleting Products

Refer to this topic for help adding and deleting additional products from your Pivotal Cloud Foundry (PCF) installation, such as Pivotal RabbitMQ® for PCF.

Note: In Ops Manager 2.5, all product tiles use floating stemcells by default. This increases the security of your deployment by enabling tiles to automatically use the latest patched version of a stemcell, but it may significantly increase the amount of time required by a tile upgrade. Review the Floating Stemcells topic for more information.

Adding and Importing Products

1. Download PCF-compatible products at Pivotal Network. If you cannot download products from Pivotal Network due to restricted network connectivity, see Installing PCF in Airgapped Environments.

2. Navigate to your Ops Manager Installation Dashboard and log in.

3. Click Import a Product.

4. Select the .pivotal file that you downloaded from Pivotal Network or received from your software distributor, then click Open. If the product is successfully added, it appears in your product list. If the product that you selected is not the latest version, the most up-to-date version will appear on your product list.

5. Add the product tile to the Installation Dashboard by clicking the green plus sign icon.

6. The product tile appears in the Installation Dashboard. If the product requires configuration, the tile appears orange. If necessary, configure the product.
7. (Optional) In the product configuration view, select the **Errands** pane to configure post-install errands or review the default settings. Post-install errands are scripts that automatically run after a product installs, before Ops Manager makes the product available for use. For more information about post-install errands, see the [Errands](#) topic.

Note: By default, Ops Manager reruns errands even if they are not necessary due to settings left from a previous install. Leaving errands checked at all times can cause updates and other processes to take longer. To prevent an errand from running, deselect the checkbox for the errand in the **Settings** tab on the product tile before installing the product.

8. Click **Review Pending Changes**, then **Apply Changes** to start installation and run post-install lifecycle errands for the product.

The **Broker Registrar** checkbox is an example of an errand available for a product. When you select this checkbox, this errand registers service brokers with the Cloud Controller and also updates any broker URL and credential values that have changed since the previous registration.
Using Pivotal Network API to Upgrade Products

Ops Manager provides a way to upgrade products by connecting your Installation Dashboard with Pivotal Network using an API token. Once you have uploaded a product, all subsequent product upgrades appear automatically in your Installation Dashboard.

Note: Using the Pivotal Network API is only available if you have access to the Internet since communication between Ops Manager and the Pivotal Network is necessary to import your products. If you are on an isolated network, do not save your API token.

1. Navigate to Pivotal Network and log in.

2. Click your user name, located in the upper top right side of the page.

3. Select Edit Profile.

4. In the Edit Profile tab, copy your API Token.

5. Navigate to your Ops Manager Installation Dashboard and log in.

6. Click your user name, located in the upper top right side of the page.

7. Select Settings.

8. In the External API Access tab, paste your API Token.

9. Click Save.

Update Existing Products

Once you save the Pivotal Network API Token to the Ops Manager Installation Dashboard, the latest versions of your existing products will appear in your Installation Dashboard. Upgrade your product to the latest version by following these instructions.

1. Locate and download the product version you want to upgrade to by clicking on the green download icon.
2. When the download is complete, refresh the page to use the product.

3. If necessary, configure the product.

4. Click **Review Pending Changes**, then **Apply Changes**.

Applying Changes to BOSH Director

You can use the Ops Manager API or the Review Pending Changes page to apply pending changes only to BOSH Director when you stage multiple products in a new installation or as part of an upgrade.

Deleting a Product

1. From the Installation Dashboard, click the trash icon on a product tile to remove that product. In the **Delete Product** dialog box that appears, click **Confirm**.

   **Note:** You cannot delete the BOSH Director product.

2. Click **Review Pending Changes**, then **Apply Changes**.

   After you delete a product, the product tile is removed from the installation and the Installation Dashboard. However, the product appears in the **Available Products** view.
Importing and Managing Stemcells

This topic explains how to use the Stemcell Library to import and stage stemcells to products.

For more conceptual information about floating stemcells and stemcell upgrades, see Floating Stemcells.

⚠️ warning: Some product releases require Xenial stemcells, as listed in Tiles Using Xenial Stemcells in PCF. If you are importing and using Xenial stemcells for the first time, see Updates for Xenial Stemcell Support.

Import and Stage a Stemcell

Download the appropriate stemcell file from Pivotal Network, then click Import Stemcell to permanently import a stemcell into Ops Manager.

To stage your stemcell to compatible products, enable the products in the Import Stemcell dialog. Click Apply Stemcell to Products to save your selections. Click Dismiss to dismiss the Import Stemcell dialog.
Choose a Stemcell Version

If you have uploaded multiple versions of a stemcell, you can use the dropdown in the Staged column to choose which version to use.

You can choose a different version until you deploy. After deployment, older stemcell versions are no longer available.

If the stemcell displays with a green checkmark and the words Latest stemcell below the Staged dropdown, the stemcell is the latest available version on your host. An outdated stemcell displays Stemcell out-of-date.
Logging In to Apps Manager

Log In as Admin User

Complete the following steps to log in to Apps Manager as the Admin user:

1. If you do not know the system domain for the deployment, then select Pivotal Application Service (PAS) Settings > Domains to locate the configured system domain.

2. Open a browser and navigate to `apps.YOUR-SYSTEM-DOMAIN`. For example, if the system domain is `system.example.com`, then point your browser to `apps.system.example.com`.

3. Log in using UAA credentials for the Admin user. To obtain these credentials, refer to PAS Credentials > UAA > Admin Credentials.

<table>
<thead>
<tr>
<th>UAA</th>
<th>VM Credentials</th>
<th>Link to Credential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin Credentials</td>
<td></td>
<td>Link to Credential</td>
</tr>
</tbody>
</table>
Managing Errands in Ops Manager

This topic describes product errands and how to configure them in Pivotal Cloud Foundry (PCF) Ops Manager.

Errands are scripts that can run at the beginning and at the end of an installed product’s availability time. You can use Ops Manager to adjust whether and when these errands run.

Product tiles include two types of errands:

- **Post-deploy errands** run after the product installs but before Ops Manager makes the product available for use. One example is an errand that publishes a newly-installed service to the Services Marketplace.
- **Pre-delete errands** run after an operator chooses to delete the product but before Ops Manager actually deletes it. One example is a clean-up task that removes all data objects used by the errand.

When you click **Review Pending Changes**, then **Apply Changes** in Ops Manager, BOSH either creates a VM for each errand that runs or co-locates errands on existing VMs. Tile developers determine where BOSH deploys the errands for their product.

Pivotal Application Service (PAS) provides several post-deploy errands including smoke tests, Apps Manager, notification, Pivotal Account, and autoscaling errands. For more information about PAS errands, see the **Configuring PAS** topic.

For information about the errands associated with any other PCF product, see the documentation provided with the product tile.

Errand Run Rules

Operators can configure two different run rules for errands: **On** and **Off**. These rules control when Ops Manager executes the errand.

When the errand is configured to be **On**, then it always runs, even if there are no changes to the product manifest. When the errand is configured to be **Off**, it never runs.

Ops Manager Defaults and Tile Defaults

By default, Ops Manager applies the **On** rule to all errands.

For any errand, the tile developer can override these Ops Manager defaults with their own tile-specific defaults defined in the tile property blueprints.

Configure Run Rules in Ops Manager

You can configure the run rules for errands in two places in Ops Manager. The **Errands** pane saves your configuration and applies the configuration to future installations. The **Pending Changes** view applies the rules only to the next time you run an Ops Manager install, without saving them.

Errands Pane: Persistent Rules

Product tiles for PAS and other PCF products have an **Errands** pane that configures the run rules for the product’s errands and saves the settings for later.

The **Errands** pane lists all errands for the product and lets you select **On** or **Off**. The **default** option differs depending on the errand, and reflects the default setting used by Ops Manager for the errand or any tile-specific default that overrides it.
To configure the run rules for a tile, do the following:

1. Navigate to Ops Manager and click the tile to open it.
2. Under the **Settings** tab, open the **Errands** pane.
3. Use the dropdowns to configure the run rule choice for each errand: **On** or **Off**.
4. Click **Save** to save the configuration values and return to the Installation Dashboard.
5. Click **Review Pending Changes**, then **Apply Changes** to redeploy the tile with the new settings.

Pending Changes: One-Time Rules

Ops Manager lets you quickly configure one-time errand run rules for any product queued up for installation:

1. Navigate to Ops Manager. The **Pending Changes** section at top right shows products that Ops Manager has yet to install or update.
2. Under **Pending Changes**, click the product you wish to configure. A list of errands associated with the product appears.

3. Use the dropdowns to configure the **run rule** choice for each errand: **On** or **Off**. Ops Manager applies these settings once you click **Review Pending Changes**, then **Apply Changes** to install the product, but does not save the settings for future installations.

4. Click **Review Pending Changes**, then **Apply Changes** to redeploy the tile.

**Related Links**

If you are a product developer and want to learn more about adding errands to your product tile for PCF, see the **Errands** topic in the **PCF Tile Developer Guide**.
Creating UAA Clients for BOSH Director

Page last updated:

This topic assumes you are using BOSH CLI v2+

This topic describes the process of creating a UAA client for the BOSH Director. You must create an automation client to run BOSH from a script or set up a continuous integration pipeline.

To create an automation client, do one of the following:

- **Local Authentication**: When using Internal Authentication, use the existing admin client to manually create an automation client with the correct privileges.
- **Provision Admin Client**: When setting up SAML or LDAP authentication, ensure Ops Manager provisions an admin client.

### Local Authentication

To perform this procedure, the UAAC client must be installed on the Ops Manager virtual machine (VM).

1. Open a terminal and SSH into the Ops Manager VM by following the instructions for your IaaS in the **SSH into Ops Manager** topic.

2. Navigate to the Ops Manager **Installation Dashboard** and select the **BOSH Director** tile. In BOSH Director, click the **Status** tab, and record the IP address.

3. Using the `uaac target` command, target BOSH Director UAA on port 8443 using the IP address you copied, and specify the location of the root certificate. The default location is `/var/tempest/workspaces/default/root_ca_certificate`

   ```
   Target: https://10.85.16.4:8443
   ```

   Where:
   - **BOSH-DIRECTOR-IP** is the IP you recorded in the **Status** tab of the BOSH Director.
   - **Note**: You can also curl or point your browser to the following endpoint to obtain the root certificate: https://OPS-MANAGER-FQDN/api/v0/security/root_ca_certificate

4. Log in to the BOSH Director UAA and retrieve the owner token. Perform the following step to obtain the values for **UAA-LOGIN-CLIENT-PASSWORD** and **UAA-ADMIN-CLIENT-PASSWORD**:
   a. Select the **BOSH Director** tile from the Ops Manager **Installation Dashboard**.
   b. Click the **Credentials** tab, and record the entries for **Uaa Login Client Credentials** and **Uaa Admin User Credentials**.
   c. For each entry, click **Link to Credential** to obtain the password.

   ```
   $ uaac token owner get login -s UAA-LOGIN-CLIENT-PASSWORD
   User name: admin
   Password: UAA-LOGIN-CLIENT-PASSWORD
   Successfully fetched token via owner password grant.
   Target: https://10.85.16.4:8443
   Context: admin, from client login
   ```

   **Note**: To obtain the password for the UAA login and admin clients, you can also curl or point your browser to the following endpoints:  
   - [https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/uaa_login_client_credentials](https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/uaa_login_client_credentials)
   - [https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/uaa_admin_user_credentials](https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/uaa_admin_user_credentials)

5. Create a new UAA Client with **bosh.admin** privileges.
$ uaa: client add ci --authorized_grant_types client_credentials
   --authorities bosh.admin --secret CI-SECRET

  scope: uaa.none
  client_id: ci
  resource_ids: none
  authorized_grant_types: client_credentials
  autoapprove:
  action: none
  authorities: bosh.admin
  name: ci
  lastmodified: 1669727130702
  id: ci

6. Set the client and secret as environment variables on the VM.

   $ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT=ci
   $ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT_SECRET=CI-SECRET

7. Set an alias for the BOSH Director environment.

   $ bosh alias-env MY-ENVIRONMENT-NAME -e BOSH-DIRECTOR-IP
       --ca-cert /var/tempest/workspaces/default/root_ca_certificate

   You can now use the UAA client you created to run BOSH in automated or scripted environments, such as continuous integration pipelines.

Provision Admin Client

Pivotal does not support LDAP or SAML authentication to the BOSH Director. Ops Manager provides an option to create UAA clients during SAML or LDAP configuration so that BOSH can be automated using scripts and tooling.

1. Select **Provision an admin client in the Bosh UAA** when configuring Ops Manager for SAML or LDAP.

2. Click the **Status** tab, and record the IP address, after deploying BOSH Director (BOSH).

3. Click the **Credentials** tab in the BOSH Director tile.

4. Click the link for the **Uaa Bosh Client Credentials** to record the client name and secret.

5. Open a terminal and SSH into the Ops Manager VM. Follow the instructions for your SSH in the [SSH into Ops Manager](#) topic.

6. Set the client and secret as environment variables on the Ops Manager VM. 
   
   export BOSH_CLIENT=bosh_admin_client
   export BOSH_CLIENT_SECRET=UAA-BOSH-CLIENT-SECRET

   Where:

   - **UAA-BOSH-CLIENT-SECRET** is the client secret you recorded in Step 4. For example:

     $ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT=bosh_admin_client
     $ ubuntu@ip-10-0-0-12:~$ export BOSH_CLIENT_SECRET=aBcDeFGhijKabdsadfdfdf

7. Set an alias for the BOSH Director environment.

   $ bosh alias-env MY-ENVIRONMENT-NAME -e BOSH-DIRECTOR-IP
       --ca-cert /var/tempest/workspaces/default/root_ca_certificate

   Where:

   - **BOSH-DIRECTOR-IP** is the IP you recorded in the **Status** tab of the BOSH Director.
Using Your Own Load Balancer

This guide describes how to use your own load balancer and forward traffic to your Pivotal Application Service (PAS) router IP address.

Pivotal Cloud Foundry (PCF) includes a tier of reverse proxies that dynamically track the location of application containers and system components, enabling routing of requests to those endpoints even as IPs and ports change.

In order for the PCF Routers to be horizontally scalable and highly available, a load balancer must be deployed in front of them. The simplest solution is to use a layer-4 TCP load balancer, provided by your IaaS or IT team, which passes all HTTP and TLS handling to the PCF Routers. For details on TLS termination, see Securing Traffic into Cloud Foundry. For a description of features supported by the PCF routing tier, see HTTP Routing.

If you have requirements that are not fulfilled by the PCF Routers alone, you can choose to use your own layer-7 load balancer (provided by your IaaS or IT team), or the HAProxy load balancer included with PCF. If you choose to use HAProxy, you must use a layer-4 TCP load balancer in front of it, in order for HAProxy itself to be highly available. Singleton instances of HAProxy are only for use in lab and test environments.

If you choose to use your own layer-7 load balancer, it must fulfill the following requirements:

- Provides load balancing to each of the PCF Router IP addresses
- Supports TLS termination for wildcard hostnames
- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers to incoming requests
- Sets an HTTP keep-alive connection timeout greater than five seconds
- (Optional) Supports WebSockets

The choice to use HAProxy or your own load balancer depends on what features you need out of a load balancer, and whether you want the ability to configure it yourself.

Note: Application logging with Loggregator requires WebSockets. To use another logging service, see the Using Third-Party Log Management Services topic.

For how to install an F5 Local Traffic Manager (LTM) as a load balancer for PCF and Pivotal Application Service (PAS), see Configuring an F5 Load Balancer for PAS.

Prerequisites

To integrate your own load balancer with PCF, you must ensure the following:

- WebSocket connections are not blocked for Loggregator functionality.
- The load balancer must be able to reach the Gorouter IP addresses.

Follow the instructions below to use your own load balancer.

Step 1: Deploy PCF Installation VM

Deploy a PCF Installation virtual machine. See Deploying Ops Manager on vSphere for more information.

Step 2: Register PCF IP Address

In your load balancer, register the IP addresses that you assigned to PCF.

Step 3: Configure Pivotal Ops Manager and BOSH Director

Configure your Pivotal Operations Manager and BOSH Director as described in Installing Pivotal Cloud Foundry, then add PAS.

Do not click Install after adding PAS.
Step 4: Configure Networking

Complete the Networking configuration page in PAS. Load balancer configuration in PAS varies depending on which IaaS you are using for PCF. See Configuring PAS.

Step 5: Finalize Changes

1. Return to the Ops Manager Installation Dashboard
2. Click Install.
Creating and Managing Ops Manager User and Client Accounts

Page last updated:

This topic describes how to add and remove Ops Manager users. It also describes how you can use an admin Ops Manager account to create client accounts that you can use for Ops Manager automation.

Overview

Pivotal Cloud Foundry supports multiple user accounts in Ops Manager. A User Account and Authentication (UAA) module co-located on the Ops Manager VM manages access permissions to Ops Manager.

When Ops Manager boots for the first time, you create an admin user. However, you do not create additional users through the Ops Manager web interface. If you want to create additional users who can log into Ops Manager, you must use the UAA API, either through `curl` or the UAA Command Line Client (UAAC).

Users are not the only type of account you can create for Ops Manager. You can also create client accounts, which connect automation tools and scripts to Ops Manager. Pivotal recommends using clients to handle automated tasks.

Client accounts are not bound to the same authentication protocols as user accounts. A user account that controls automated components can cause those components to fail if the account experiences inconsistent availability due to permission or authentication issues.

You can create client accounts after deploying Ops Manager, or during configuration for an initial deployment. For more information about adding clients during initial configuration or after deployment, see Add Pre-Created Client.

Add Ops Manager Users

This section describes how to add or remove users with UAAC. If you do not already have the UAAC installed, run `gem install cf-uaac` on the command line.

Note: You can only manage users on the Ops Manager UAA module if you chose to use Internal Authentication instead of an external Identity Provider when configuring Ops Manager.

To add Ops Manager users, do the following:

1. Target your Ops Manager UAA:
   
   ```
   uaac target https://YOUR-OPSMANAGER-FQDN/uaa/
   ```

   Where:
   
   - `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.

2. Get your token:
   
   ```
   uaac token owner get
   ```

   Client ID: opsmn
   
   Client Secret: 
   
   Username: Admin
   
   Password: ********
   
   Successfully fetched token via client credentials grant.
   
   Target https://YOUR-OPSMANAGER-FQDN/uaa

   Where:
   
   - `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.

3. Add a user:
   
   ```
   uaac user add USER-NAME -p USER-PASSWORD --emails USER-EMAIL@EXAMPLE.COM
   ```

   Where:
Remove Ops Manager Users

To remove Ops Manager users, do the following:

1. Target your Ops Manager UAA:

   ```
uaac target https://YOUR-OPSMANAGER-FQDN/uaa/
   ```

2. Get your token:

   ```
uaac token-owner get
   Client ID: opsman
   Client Secret: 
   Username: Admin
   Password: ******
   ```

   Successfully fetched token via client credentials grant.
   Target https://YOUR-OPSMAN-FQDN/uaa/

   Where:

   - `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.

3. Delete a user:

   ```
uaac user delete USER-NAME
   ```

   Where:

   - `USER-NAME` is the username of the user you wish to delete.

Add Ops Manager Client Accounts

The following sections describe how to create client accounts for Ops Manager automation using an admin account.

Log in to UAAC as an Admin

In order to configure a client, you must first log in to UAAC as an admin.

Use one of the following two methods to authenticate to UAAC:

- Authenticate Using SAML or SSO
- Authenticate Using LDAP

Authenticate Using SAML or SSO

If you’re using SAML or SSO, authenticate to UAAC as an admin before creating a client.

To authenticate to UAAC, do the following:

1. Target your UAA server.

   ```
uaac target https://YOUR-OPSMANAGER-FQDN/uaa
   ```
Where:
- `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.

2. Log in as an admin.
   
   `uaac token sso get`

3. When prompted, type the Client ID and passcode. Leave the client secret blank.

<table>
<thead>
<tr>
<th>Client ID: opsman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client secret:</td>
</tr>
<tr>
<td>Passcode (from [<a href="http://YOUR-OPSMANAGER-FQDN/uaa/passcode">http://YOUR-OPSMANAGER-FQDN/uaa/passcode</a>]): YOUR-UAA-PASSCODE</td>
</tr>
</tbody>
</table>

Where:
- `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.
- `YOUR-UAA-PASSCODE` is the passcode with which you authenticate to UAA.

### Authenticate Using LDAP

If you’re not using SAML or SSO, authenticate to UAAC as an admin before creating a client.

To authenticate to UAAC, do the following:

1. Target your UAA server.
   
   `uaac target https://YOUR-OPSMANAGER-FQDN/uaa`

   Where:
   - `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.

2. Log in as an admin.
   
   `uaac token owner get`

3. When prompted, type the Client ID, your username, and your password. Leave the client secret blank.

<table>
<thead>
<tr>
<th>Client ID: opsman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client secret:</td>
</tr>
<tr>
<td>User name: admin</td>
</tr>
<tr>
<td>Password: *****</td>
</tr>
</tbody>
</table>

4. A message appears confirming that UAAC has [Successfully fetched token].

### Create a Client

After you authenticate to UAAC, create a client to manage automated components and tasks.

To create a client, do the following:

1. Create a client with role-based permissions, an ID, and an authentication secret using UAAC:
   
   `uaac client add CLIENT-ID --authorized_grant_types client_credentials --authorities opsman.admin --secret CLIENT-SECRET`

   Where:
   - `CLIENT-ID` is the name of your client.
   - `CLIENT-SECRET` is the secret you use to authenticate to your client.

   The `opsman.admin` authority referenced in the example above grant specific permissions to the client. You can choose which permissions you wish the client to have by assigning a different role.
For more information about Ops Manager roles and what they do, see [Understanding Roles in Ops Manager](#).

## Add Pre-Created Client

You can add a pre-created client to Ops Manager using the Ops Manager API in either of the following ways:

- Add a Pre-Created Client Before Initial Deployment
- Add or Modify a Pre-Created Client for an Existing Deployment

### Add a Pre-Created Client Before Initial Deployment

This section describes how to add a pre-created client before you deploy Ops Manager for the first time.

If you do not already have the UAAC installed, run `gem install cf-uaac` on the command line.

To add a pre-created client before you deploy Ops Manager, do the following:

1. Authenticate to and access the Ops Manager API by following the steps in [Using the Ops Manager API](#).
2. Pass the following command:
   ```
   POST /api/v0/setup
   ```
   Include any configuration details your deployment requires. For more information about configuring Ops Manager with the API, see the [Setting Up with SAML](#) in the Ops Manager API documentation.
3. In the configuration details, specify this parameter:
   ```
   "precreated_client_secret": "YOUR-SECRET"
   ```
   Where:
   - **YOUR-SECRET** is the authentication secret you use to access the pre-created client.
   
   ![Note](https://example.com) The authentication secret for the pre-created client must be between 1 and 255 ASCII characters.

   When `precreated_client_secret` is passed, a UAA client is created called `precreated-client`. To use this client, see [Authenticate as the Pre-Created Client](#).

### Add or Modify a Pre-Created Client for an Existing Deployment

This section describes how to add a pre-created client after you have already deployed Ops Manager. You may only have one pre-created client at a time. If you have an existing pre-created client, you may also use this procedure to change its authentication secret.

To add a pre-created client to an Ops Manager that has already been deployed, or change the secret of an existing pre-created client, do the following:

1. Authenticate to and access the Ops Manager API by following the steps in [Using the Ops Manager API](#).
2. Pass the following command:
   ```
   PUT /api/v0/uaa/precreated_client
   "precreated_client_secret": "NEW-SECRET"
   ```
   Where:
   - **NEW-SECRET** is the authentication secret you use to access the pre-created client. This secret must differ from the previous secret you used. The secret cannot be blank.

   ![Note](https://example.com) The authentication secret for the pre-created client must be between one and 255 ASCII characters.
To use this client, see Authenticate as the Pre-Created Client

Authenticate as the Pre-Created Client

To authenticate to UAAC as the pre-created client, do the following:

1. Target your UAA server.
   
   ```bash
   uaac target https://YOUR-OPSMANAGER-FQDN/uaa
   ```
   
   Where:
   
   - `YOUR-OPSMANAGER-FQDN` is the fully qualified domain name of your Ops Manager installation.

2. Log in as a client.
   
   ```bash
   uaac token client get precreated-client -s "PRECREATED-CLIENT-SECRET"
   ```

3. A message appears confirming that UAAC has successfully fetched token.

You have authenticated to UAAC as the pre-created client. Use this client to perform any automation tasks you wish.

For more information about the Ops Manager API, see Using the Ops Manager API.
Configuring Role-Based Access Control (RBAC) in Ops Manager

This topic describes how to customize role-based access control (RBAC) in Ops Manager. Use RBAC to manage which operators in your organization can make deployment changes, view credentials, and manage user roles in Ops Manager.

For information about configuring Ops Manager to use internal authentication or SAML authentication, see the Ops Manager configuration topic for your IaaS:

- Configuring BOSH Director on AWS
- Configuring BOSH Director on Azure Manually
- Configuring BOSH Director on GCP
- Configuring BOSH Director on OpenStack
- Configuring BOSH Director on vSphere

Roles in Ops Manager

You can assign the following roles to determine which operators in your organization make deployment changes, view credentials, and manage user roles in Ops Manager:

Ops Manager administrators can use the roles defined in the diagram above to meet the security needs of their organization. The roles provide a range of privileges that are appropriate for different types of users. For example, assign either Restricted Control or Restricted View to an operator to prevent access to all Ops Manager credentials.

See the following table for more information about each role:

<table>
<thead>
<tr>
<th>Ops Manager Role</th>
<th>Role Definition</th>
<th>UAA Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager Administrator</td>
<td>Administrators can make configuration changes and click Review Pending Changes and Apply Changes in Ops Manager, view credentials in the Credentials tab and Ops Manager API endpoints, change the authentication method, and assign roles to other operators.</td>
<td>opsman.ad min</td>
</tr>
</tbody>
</table>
Full Control
Operators can make configuration changes and click Review Pending Changes and Apply Changes in Ops Manager, and view credentials in the Credentials tab and Ops Manager API endpoints.

Restricted Control
Operators can make configuration changes and click Review Pending Changes and Apply Changes in Ops Manager. They cannot view credentials in the Credentials tab or Ops Manager API endpoints.

Full View
Operators can view Ops Manager configuration settings and view credentials in the Credentials tab and Ops Manager API endpoints. They cannot make configuration changes or click Apply Changes.

Restricted View
Operators can view Ops Manager configuration settings. They cannot make configuration changes or view credentials in the Credentials tab or Ops Manager API endpoints.

To assign one of the above roles to an operator, follow the procedure for granting access using either internal authentication or SAML authentication.

When you install a new Ops Manager instance, all existing users have the Ops Manager Administrator role by default.

Simultaneous Ops Manager Administrators
Ops Manager allows multiple administrators to log in to Ops Manager simultaneously and make changes.

The interface does not provide visibility to other administrators that are logged in. Pivotal recommends that administrators communicate with each other and coordinate their changes.

Precedence for Apply Changes
Only one deployment takes precedence when two administrators try to deploy around the same time.

If two administrators are working at the same time, the administrator who first clicks Apply Changes takes precedence. Ops Manager overwrites all configurations made by other administrators during deployment.

Pivotal recommends coordinating changes between administrators to avoid overwriting configurations.

Enable RBAC in Ops Manager After Upgrade
When you install a new instance of Ops Manager, RBAC is permanently enabled by default.

If your organization has operators who are devoted to managing certain services like MySQL for PCF, you can use RBAC to assign those services operators a more restricted role.

If you upgrade from an older Ops Manager instance, you must enable RBAC and assign roles to users before they can access Ops Manager. If you do not assign any roles to a user, they cannot log in to Ops Manager.

⚠️ warning: Do not assign roles before you enable RBAC.

Enable RBAC with Internal Authentication
If you are upgrading from an older version of Ops Manager and use internal authentication, do the following to enable RBAC:

1. Log in to the Ops Manager dashboard.
2. Click Settings from the user account menu.
3. Click Advanced.
4. Click Enable RBAC. When the confirmation dialog box appears, click Confirm and Logout.

© Copyright Pivotal Software Inc, 2013-2019
Enable RBAC with SAML Authentication

If you are upgrading from an older version of Ops Manager and use SAML authentication, perform the steps in this section to enable RBAC. To enable RBAC in Ops Manager when using SAML authentication, you must configure groups in SAML for admins and non-admins and then map the admin group to Ops Manager.

Step 1: Configure SAML Groups
To gather information from your SAML dashboard, do the following:

1. Log in to your SAML provider dashboard.
2. Create or identify the name of the SAML group that contains Ops Manager admin users.
3. Identify the groups attribute tag you configured for your SAML server.

Step 2: Enable RBAC in Ops Manager
Perform the steps above in Enable RBAC with Internal Authentication to configure Ops Manager to recognize your SAML admin user group.

Note: When RBAC is enabled, only users with the Ops Manager Administrator role can edit SAML configuration.

Create User Accounts in Ops Manager
To assign RBAC roles to operators, you must first create user accounts for them. For more information about creating user accounts in Ops Manager with the User Account and Authentication (UAA) module, see Creating and Managing Ops Manager User Accounts.

In addition to user accounts, you can create a client account to add to Ops Manager. Client accounts manage automation tasks, such as upgrade scripts, log management, and other behaviors that might be negatively impacted if managed by a user account. You can add a client account either before initial deployment or to an existing deployment.

For more information about client accounts, see Creating and Managing Ops Manager User and Client Accounts.

Manage RBAC Roles in Ops Manager
You can assign the roles defined in Roles in Ops Manager to determine which operators in your organization make deployment changes, view credentials, and manage user roles in Ops Manager.

Manage Roles with Internal Authentication
If you configured Ops Manager to use internal authentication, do the following to configure roles using the UAA Command Line Interface (UAAC):

1. Target your UAA server and log in as an admin:

```
uaac target https://YOUR-OPSMAN-DOMAIN/uaa
uaac token owner get
```

2. When prompted, enter the following credentials. Enter opsman for Client ID and leave Client secret blank, then enter your username and

Notes:
- Enabling RBAC is permanent. You cannot undo this action. When you upgrade Ops Manager, your RBAC settings remain configured.
- You will not see this dialog box if RBAC is already configured. With new instances of Ops Manager, RBAC is permanently configured by default.

© Copyright Pivotal Software Inc, 2013-2019
3. Assign one of the following roles to a user, replacing `USERNAME` with their username.

- **Ops Manager Administrator:**
  
  `uaac member add opsman.admin USERNAME`

- **Full Control:**
  
  `uaac member add opsman.full_control USERNAME`

- **Restricted Control:**
  
  `uaac member add opsman.restricted_control USERNAME`

- **Full View:**
  
  `uaac member add opsman.full_view USERNAME`

- **Restricted View:**
  
  `uaac member add opsman.restricted_view USERNAME`

**Manage Roles with SAML Authentication**

If you configured Ops Manager with SAML authentication, do the following to assign non-admin user roles using UAAC:

1. Target your UAA server and log in as an admin:

   ```
   uaac target https://YOUR-OPSMAN-DOMAIN/uaa
   uaac token sso get
   ```

2. When prompted, enter **Client ID** and **Passcode**, leaving **Client secret** blank:

   ```
   Client ID: opsman
   Client secret: 
   ```

3. Run the following command:

   ```
   uaac group map SAML-GROUP --name 'OPSMAN-SCOPE' --origin 'saml'
   ```

   Replace the placeholder text as follows:

   - **SAML-GROUP** : Replace with name of the SAML group the user belongs to.
   - **OPSMAN-SCOPE** : Replace with an Ops Manager UAA scope. See the table in [Understand Roles in Ops Manager](#) to determine which UAA scope to use.

4. Add new and existing users to the appropriate SAML groups in the SAML provider dashboard. Users must log out of both Ops Manager and the SAML provider for role changes to take effect.
Creating New PAS User Accounts

When you first deploy your Pivotal Application Service (PAS), there is only one user: an administrator. At this point, you can add accounts for new users who can then push applications using the Cloud Foundry Command Line Interface (cf CLI).

How to add users depends on whether or not you have SMTP enabled, as described in the options below.

Option 1: Adding New Users when SMTP is Enabled

If you have enabled SMTP, your users can sign up for accounts and create their own orgs. They do this using the Pivotal Cloud Foundry (PCF) Apps Manager, a self-service tool for managing organizations, users, applications, and application spaces.

Instruct users to complete the following steps to log in and get started using the Apps Manager.

1. Browse to `apps.YOUR-SYSTEM-DOMAIN`. Refer to PAS Domains to locate your system domain.
2. Select Create an Account.
3. Enter your email address and click Create an Account. You will receive an email from the Apps Manager when your account is ready.
4. When you receive the new account email, follow the link in the email to complete your registration.
5. You will be asked to choose your organization name.

You now have access to the Apps Manager. Refer to the Apps Manager documentation at [docs.pivotal.io](https://docs.pivotal.io) for more information about using the Apps Manager.

Option 2: Adding New Users when SMTP is Not Enabled

If you have not enabled SMTP, only an administrator can create new users, and there is no self-service facility for users to sign up for accounts or create orgs.

The administrator creates users with the cf CLI. See Creating and Managing Users with the cf CLI.

Return to the Installing Pivotal Cloud Foundry Guide.
Adding Existing SAML or LDAP Users to a PCF Deployment

This topic describes the procedure for adding existing SAML or LDAP users to a Pivotal Cloud Foundry (PCF) deployment enabled with SAML or LDAP.

The following two ways exist to add existing SAML or LDAP users to your PCF deployment:

- **Option 1: Import Users in Bulk**
- **Option 2: Add Users Manually**

Prerequisites

You must have the following to perform the procedures in this topic:

- Admin access to the Ops Manager Installation Dashboard for your PCF deployment
- The Cloud Foundry Command Line Interface (cf CLI) v6.23.0 or later

Option 1: Import Users in Bulk

You can import SAML or LDAP users in bulk by using the UAA Bulk Import Tool. See the UAA Users Import README for instructions about installing and using the tool.

Option 2: Add Users Manually

Perform the procedures below to add existing SAML or LDAP users to your PCF deployment manually.

Step 1: Create User

Perform the following steps to add a SAML or LDAP user:

1. Run `cf target https://api.YOUR-SYSTEM-DOMAIN` to target the API endpoint for your PCF deployment. Replace `YOUR-SYSTEM-DOMAIN` with your system domain. For example:

   ```
   $ cf target https://api.example.com
   ```

2. Run `cf login` and provide credentials for an account with the Admin user role:

   ```
   $ cf login
   ```

3. Run `cf create-user EXAMPLE-USERNAME --origin YOUR-PROVIDER-NAME` to create the user in UAA. Replace `EXAMPLE-USERNAME` with the username of the SAML or LDAP user you wish to add, and select one of the options below:

   - For LDAP, replace `YOUR-PROVIDER-NAME` with `ldap`. For example:
     ```
     $ cf create-user j.smith@example.com --origin ldap
     ```

   - For SAML, replace `YOUR-PROVIDER-NAME` with the name of the SAML provider you provided when configuring Ops Manager. For example:
     ```
     $ cf create-user j.smith@example.com --origin example-saml-provider
     ```

Step 2: Associate User with Org or Space Role

After creating the SAML or LDAP user, you must associate the user with either an Org or Space role.

For more information about roles, see the Roles and Permissions section of the Orgs, Spaces, Roles, and Permissions topic.
Associate User with Org Role

Run `cf set-org-role USERNAME YOUR-ORG ROLE` to associate the SAML or LDAP user with an Org role. Replace `USERNAME` with the name of the SAML or LDAP user, and replace `YOUR-ORG` with the name of your Org.

For `ROLE`, enter one of the following:

- **OrgManager**: Org Managers can invite and manage users, select and change plans, and set spending limits.
- **BillingManager**: Billing Managers can create and manage the billing account and payment information.
- **OrgAuditor**: Org Auditors have read-only access to Org information and reports.

Example:

```
$ cf set-org-role j.smith@example.com my-org OrgManager
```

Associate User with Space Role

Run `cf set-space-role USERNAME YOUR-ORG YOUR-SPACE ROLE` to associate the SAML or LDAP user with a Space role. Replace `USERNAME` with the name of the SAML or LDAP user, replace `YOUR-ORG` with the name of your Org, and `YOUR-SPACE` with the name of a Space in your Org.

For `ROLE`, enter one of the following:

- **SpaceManager**: Space Managers can invite and manage users, and enable features for a given Space.
- **SpaceDeveloper**: Space Developers can create and manage apps and services, and see logs and reports.
- **SpaceAuditor**: Space Auditors can view logs, reports, and settings on this Space.

Example:

```
$ cf set-space-role j.smith@example.com my-org my-space SpaceDeveloper
```
Modifying Your Ops Manager Installation and Product Template Files

This topic describes how to modify your Ops Manager installation by decrypting and editing the YAML files that Ops Manager uses to store configuration data. Operators can use these procedures to view and change values that they cannot access through the Ops Manager web interface. They can also modify the product templates that Ops Manager uses to create forms and obtain user input.

Operators may want to modify the Ops Manager installation and product template files for a number of reasons, including the following:

- To change the User Account and Authentication (UAA) admin password of their deployment
- To retrieve key values
- To migrate content across different Pivotal Cloud Foundry (PCF) releases

⚠️ warning: Be careful when making changes to your Ops Manager installation and product template files. Use spaces instead of tabs, and remember that YAML files use whitespace as a delimiter. Finally, Pivotal does not officially support these procedures, so use them at your own risk.

Understand Installation and Product Template Files

During the installation process, Ops Manager combines information from the installation and product template files to generate the manifests that define your deployment.

- **Installation files**: PCF stores user-entered data and automatically generated values for Ops Manager in installation YAML files on the Ops Manager virtual machine (VM). PCF encrypts and stores these files in the directory `/var/tempest/workspaces/default`. You must decrypt the files to view their contents, edit them as necessary, then re-encrypt them.

- **Product templates**: Ops Manager uses product templates to create forms and obtain user input. The `job_types` and `property_blueprint` key-value pairs in a product template determine how the `jobs` and `properties` sections display in the installation file. Ops Manager stores product templates as YAML files in the directory `/var/tempest/workspaces/default/metadata` on the Ops Manager VM. These files are not encrypted, so you can edit them without decrypting. User input does not alter these files.

💡 Note: Upgrading Ops Manager may eliminate your changes to the installation and product template files.

Modify the Installation Files

Perform the following steps to locate, decrypt, and edit the Ops Manager installation files `installation.yml` and `actual-installation.yml`.

1. SSH into the Ops Manager VM by following the steps in the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic.

2. On the command line, navigate to the scripts directory:

   ```bash
   $ cd /home/tempest-web/tempest/web/scripts/
   ```

3. Run the following command to decrypt the installation YAML file and make a temporary copy of the decrypted file. When prompted for a passphrase, enter the decryption passphrase you created when you launched Ops Manager for the first time:

   ```bash
   $ sudo -u tempest-web ./decrypt /var/tempest/workspaces/default/installation.yml /tmp/installation.yml
   ```

4. Open `/tmp/installation.yml` to view or edit values.

5. If you plan to make changes, make a backup of the original installation YAML file:

   ```bash
   $ cp /var/tempest/workspaces/default/installation.yml ~/.installation-orig.yml
   ```

6. If you have made changes to your copy of the installation YAML file, you must encrypt it and overwrite the original with it:

   ```bash
   ```
When prompted, enter a passphrase.

7. Delete the temporary copy of the decrypted file:

   $ rm /tmp/installation.yml

8. Repeat steps 2 through 7 for `/tmp/actual-installation.yml`. Each step you see `installation.yml`, replace with `actual-installation.yml`. For example, for step 3 you run:

   $ sudo -u tempest-web /decrypt /var/tempest/workspaces/default/actual-installation.yml /tmp/actual-installation.yml

9. Restart the Ops Manager web interface:

   $ sudo service tempest-web stop && sudo service tempest-web start

10. Navigate to Ops Manager in a browser and enter your decryption passphrase.

11. Log in to Ops Manager.

12. Click Review Pending Changes, then Apply Changes.

13. If Ops Manager cannot load your changes, see the Revert To Your Backup section of this topic to restore your previous settings.

Modify Product Template Files

Perform the following steps to locate and edit your Ops Manager product template files:

1. SSH into the Ops Manager VM by following the steps in the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic.

2. On the Ops Manager VM, navigate to the `/var/tempest/workspaces/default/metadata` directory.

   $ cd /var/tempest/workspaces/default/metadata

3. The `/var/tempest/workspaces/default/metadata` directory contains the product templates as YAML files. If you plan to make changes, make a backup of the original product template YAML file:

   $ cp /var/tempest/workspaces/default/metadata/YOUR-PRODUCT-TEMPLATE.yml ~/YOUR-PRODUCT-TEMPLATE-orig.yml

4. Open and edit the product template YAML file as necessary. For more information about product templates, see the Product Template Reference topic.

5. Navigate to Ops Manager to see your changes.

6. If Ops Manager cannot load your changes, see the Revert To Your Backup section of this topic to restore your previous settings.

Revert to Your Backup

Perform the following steps to revert to your backup of an installation or product template file:

1. SSH into the Ops Manager VM by following the steps in the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic.

2. Overwrite the modified file with the backup:

   - For the installation file, run the following command:
     
     $ cp ~/installation-orig.yml /var/tempest/workspaces/default/installation.yml
   
   - For a product template file, run the following command:
     
     $ cp ~/YOUR-PRODUCT-TEMPLATE-orig.yml /var/tempest/workspaces/default/metadata/YOUR-PRODUCT-TEMPLATE.yml
3. Restart the Ops Manager web interface:

```
$ sudo service tempest-web stop && sudo service tempest-web start
```

4. Navigate to Ops Manager in a browser and enter your decryption passphrase.

5. Log in to Ops Manager.

6. Click **Review Pending Changes**, then **Apply Changes**.
Monitoring PCF VMs from Ops Manager

This topic describes how to check current VM status in Pivotal Cloud Foundry (PCF) Ops Manager.

For a complete guide to monitoring PCF, see Monitoring Pivotal Cloud Foundry.

Monitoring VMs Using the Ops Manager Interface

Click any product tile and select the Status tab to view monitoring information.

![Image of Ops Manager interface with monitoring information]

The columns display the following information:

<table>
<thead>
<tr>
<th>VM Data Point</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job</td>
<td>Each job represents a component running on one or more VMs that Ops Manager deployed.</td>
</tr>
<tr>
<td>Index</td>
<td>For jobs that run across multiple VMs, the index value indicates the order in which the job VMs were deployed. For jobs that run on only one VM, the VM has an index value of 0.</td>
</tr>
<tr>
<td>IPs</td>
<td>IP address of the job VM.</td>
</tr>
<tr>
<td>CID</td>
<td>Uniquely identifies the VM.</td>
</tr>
<tr>
<td>Load Avg15</td>
<td>CPU load average over 15 minutes.</td>
</tr>
<tr>
<td>CPU</td>
<td>Current CPU usage.</td>
</tr>
<tr>
<td>Memory</td>
<td>Current memory usage.</td>
</tr>
<tr>
<td>Swap</td>
<td>Swap file percentage.</td>
</tr>
<tr>
<td>System Disk</td>
<td>System disk space usage.</td>
</tr>
<tr>
<td>Ephem. Disk</td>
<td>Ephemeral disk space usage.</td>
</tr>
<tr>
<td>Pers. Disk</td>
<td>Persistent disk space usage.</td>
</tr>
<tr>
<td>Logs</td>
<td>Download link for the most recent log files.</td>
</tr>
</tbody>
</table>

Operations Manager VM Disk Space

The Ops Manager stores its logs on the Ops Manager VM in the /tmp directory.
Note: The logs collect over time and do not self-delete. To prevent the VM from running out of disk space, restart the VM to clear the log entries from /tmp.
Advanced Features in Ops Manager

This guide describes advanced features for Ops Manager operators. Use these features to perform functions in Ops Manager that are outside the scope of typical Ops Manager behavior.

⚠️ warning: Misuse of any advanced feature could disable or destroy deployments. These features are for advanced operators only. Pivotal recommends contacting Pivotal Support before using these features.

In This Guide

This guide includes the following topics:

- Managing Ops Manager Verifiers
- Managing Custom VM Extensions
- Ops Manager Fields That Lock On Deploy
- Property and Template References
Managing Ops Manager Verifiers

This topic describes how to disable and re-enable verifiers for fields in the BOSH Director and other product tiles.

Ops Manager provides this option for troubleshooting purposes. For example, your deployment may have a unique configuration that the verifier cannot detect. In this case, you can unblock your deployment by disabling the verifier.

**Note:** Pivotal recommends contacting support before you disable Ops Manager verifiers.

**Warning:** Pivotal recommends recording and routinely revisiting which verifiers you disable. The Ops Manager UI does not indicate which verifiers you disable. Disabled verifiers persist through upgrades.

About Verifiers

Ops Manager verifiers narrow the feedback loop for operators by checking that fields are configured correctly. Ops Manager verifiers reach out and find objects in the world. For example, given an IP, a verifier can ping the IP to see that it responds.

Verifiers run early in a deployment. With verifiers turned off, deployments may fail much later.

Verifiers run on two occasions:

<table>
<thead>
<tr>
<th>Verifier Label</th>
<th>Triggers when you…</th>
<th>Blocks deployment?</th>
</tr>
</thead>
<tbody>
<tr>
<td>form</td>
<td>Click Save in a pane.</td>
<td>Does not block deployment. Error message is a warning.</td>
</tr>
<tr>
<td>install_time</td>
<td>Click Apply Changes.</td>
<td>Blocks you from deploying. Disabling verifiers unblocks deployment.</td>
</tr>
</tbody>
</table>

Disable Verifiers by Type

You can disable verifiers by verifier type. For example, you can disable all MySQL verifiers for a tile. To disable a verifier type for a tile, you must find and record the type, then use the Ops Manager API.

Step 1: Record the Verifier Type

To record the verifier type, do the following:

1. Log in to Ops Manager and go to Review Pending Changes.
2. Click Apply Changes and see the verifier error message.
3. Record the type in the verifier error message. For example, `WildcardDomainVerifier`.

Step 2: Use the Ops Manager API

To disable verifiers, you must use the Ops Manager API. To get started with the Ops Manager API, follow the procedures in Using the Ops Manager API.

The API endpoint varies by tile. Go to one of the procedures below:
Disable Verifiers for the BOSH Director Tile

To disable a verifier type for the BOSH Director, do the following:

1. Run the following API command:

```shell
curl "https://OPS-MAN-FQDN/api/v0/staged/director/verifiers/install_time/VERIFIER-TYPE" \
-X PUT \
-H "Authorization: Bearer UAA-ACCESS-TOKEN" \
-H "Content-Type: application/json" \
-d '{
  "enabled": false
}'
```

Where:
- **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
- **VERIFIER-TYPE** is the verifier type you recorded in **Step 1: Record the Verifier Type**.
- **UAA_ACCESS_TOKEN** is your UAA access token. For more information, see **Access the API**.

For more information about this endpoint, see **Turning off a verifier for the director** in the Ops Manager API documentation.

2. To confirm your verifier type is disabled, run the following command:

```shell
curl "https://OPS-MAN-FQDN/api/v0/staged/director/verifiers/install_time" \
-X GET \
-H "Authorization: Bearer UAA-ACCESS-TOKEN"
```

See that the Ops Manager API shows **true** for the verifier type you disabled. In the following example, **DirectorConfigurationVerifier** is set to **false**:

```
{
  "verifiers": [
    {
      "type": "AllocatedIpAddressVerifier",
      "enabled": true
    },
    {
      "type": "AvailabilityZonesVerifier",
      "enabled": true
    },
    {
      "type": "DirectorConfigurationVerifier",
      "enabled": false
    },
    {
      "type": "IaasConfigurationVerifier",
      "enabled": true
    },
    {
      "type": "NetworksExistenceVerifier",
      "enabled": true
    },
    {
      "type": "NetworksPingableVerifier",
      "enabled": true
    },
    {
      "type": "PrivilegeVerifier",
      "enabled": true
    }
  ]
}
```

For more information about this endpoint, see **Retrieving a list of install time verifiers for a specific product** in the Ops Manager API documentation.

3. Record the product and the verifier type that you disabled in a secure location. Pivotal recommends recording and routinely revisiting which verifiers you disable.

Disable Verifiers for Other Tiles

To disable a verifier type for tiles other than the BOSH Director, do the following:

1. Run the following command and record the **guid** corresponding to your product.

```shell
curl "https://OPS-MAN-FQDN/api/v0/staged/products" \
-X GET \
-H "Authorization: Bearer UAA_ACCESS_TOKEN"
```

Where:
- **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
- **UAA_ACCESS_TOKEN** is your UAA access token. For more information, see **Access the API**.

© Copyright Pivotal Software Inc, 2013-2019 704
See the following example output:

```json
[
  {
    "installation_name": "component-type1-installation-name",
    "guid": "component-type1-guid",
    "type": "component-type1"
  },
  {
    "installation_name": "p-bosh-installation-name",
    "guid": "p-bosh-guid",
    "type": "p-bosh"
  }
]
```

For more information about this endpoint, see [Listing all staged products](#) in the Ops Manager API documentation.

2. To disable your verifier by type, run the following command.

```bash
  "enabled": false
}' https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/verifiers/install_time/VERIFIER-TYPE
```

Where:

- **OPS-MAN-FQDN** is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
- **PRODUCT-GUID** is the GUID you recorded in the previous step.
- **VERIFIER-TYPE** is the verifier type you recorded in Step 1: Record the Verifier Type.

For more information about this endpoint, see [Turning off a verifier for a specific product](#) in the Ops Manager API documentation.

3. To confirm your verifier type is disabled, run the following command.

```bash
curl https://OPS-MAN-FQDN/api/v0/staged/products/PRODUCT-GUID/verifiers/install_time
-X GET -H "Authorization: Bearer UAA_ACCESS_TOKEN"
```

See that the Ops Manager API shows `false` for the verifier type you disabled. In the following example, `MysqlDatabaseVerifier` is set to `false`:

```
{
  "verifiers": [
    {
      "type": "BlobstoreVerifier",
      "enabled": true
    },
    {
      "type": "MysqlDatabaseVerifier",
      "enabled": false
    },
    {
      "type": "SSOUrlVerifier",
      "enabled": true
    },
    {
      "type": "WildcardDomainVerifier",
      "enabled": true
    }
  ]
}
```

For more information about this endpoint, see [Retrieving a list of install time verifiers for a specific product](#) in the Ops Manager API documentation.

4. Record the product and the verifier type that you disabled in a secure location. Pivotal recommends recording and routinely revisiting which verifiers you disable.

Re-enable Verifiers by Type

Pivotal recommends recording and routinely revisiting which verifiers you disable. The Ops Manager UI does not indicate which verifiers you disable. Disabled verifiers persist through upgrades.

To re-enable verifiers, follow the steps detailed in [Use the Ops Manager API](#) above. Instead of setting `enabled` to `false`, set it to `true`.

If you do not know what verifiers are disabled, `curl` the `verifiers` endpoint for each installed product. For more information, see [Retrieving a list of install time verifiers for a specific product](#) in the Ops Manager API documentation.
Managing Custom VM Extensions

This topic describes how to add, delete, apply, and list custom VM extensions using the Ops Manager API.

VM extensions allow you to specify IaaS-specific configurations for your VMs such as associated security groups and load balancers. You can use the Ops Manager API to create custom VM extensions, give them names, and associate those names with BOSH jobs.

⚠️ **warning:** If you use custom VM extensions, you may override other settings. For example, if you use a custom VM extension to add tags, the default tags are removed from all instance groups. To use default tags with custom VM extensions, apply the tags manually.

Prerequisites

To manage custom VM extensions, you need the following:

- **Access to the Ops Manager API:** For more information about the Ops Manager API, see [Using the Ops Manager API](#).
- **Your UAA Access Token Value:** Follow the procedure in [Access the API](#) in the Using the Ops Manager API topic to retrieve your access token. In each example, replace `UAA_ACCESS_TOKEN` with the value of your access token.
- **Knowledge of your IaaS-specific configurations:** To find configurations you can set for custom VM extensions, see the `VM Types` section of your IaaS CPI in the [BOSH documentation](#). For example, see [VM Types](#) in the AWS CPI docs for a list of AWS VM extension options.

Create or Update a VM Extension

From the `/api/v0/staged/vm_extensions/NAME` endpoint, use PUT to enter your custom VM extension. Replace `NAME` in the endpoint with the name of the VM extension.

Enter the following properties:

- **name:** Enter a memorable string value as the name of your new VM extension. You later associate this name to a job.
- **cloud_properties:** Enter your IaaS-specific VM extension properties.

💡 **Note:** If you are updating a custom VM extension rather than creating a new one, enter the name of the VM extension you want to override.

See the following AWS example:

```bash
curl "https://example.com/api/v0/staged/vm_extensions/example_vm_extension_name"
  -X PUT
  -H "Authorization: Bearer UAA_ACCESS_TOKEN"
  -H "Content-Type: application/json"
  -d '{"name": "example_vm_extension_name", "cloud_properties": {"source_dest_check": false}}'
```

For more information about this command, see [Creating a new vm extension](#) in the Ops Manager API documentation.

Delete a VM Extension

From the `/api/v0/staged/vm_extensions/NAME` endpoint, use DELETE to delete a custom VM extension by `NAME`. Replace `NAME` in the API endpoint with the name of the VM extension.

See the following example:

```bash
curl "https://example.com/api/v0/staged/vm_extensions/vm_ext1"
  -X DELETE
  -H "Authorization: Bearer UAA_ACCESS_TOKEN"
  -H "Content-Type: application/json"
  -d '{"name": "vm_ext1"}"
```

For more information about this command, see [Deleting a vm extension](#) in the Ops Manager API docs.
Apply VM Extensions to a Job

From the `/api/v0/staged/products/PRODUCT_GUID/jobs/JOB_GUID/resource_config` endpoint use `PUT` to configure a job with the names of your VM extensions.

For the API endpoint, replace `PRODUCT_GUID` with the Globally Unique Identifier (GUID) of your job. You can find this value by requesting

```
GET /api/v0/staged/products
```

Also replace `JOB_GUID` with the GUID of your job. You can find this value by requesting

```
GET /api/v0/staged/products/PRODUCT_GUID/jobs
```

For the `additional_vm_extensions` property, list the names of each of the custom VM extensions you want to associate with that job.

See the following GCP example:

```
curl "https://example.com/api/v0/staged/products/product-type1-guid/jobs/example-job-guid/resource_config" \
-X PUT \
-H "Authorization: Bearer UAA_ACCESS_TOKEN" \
-H "Content-Type: application/json" \
-d '{
  "instances": 1,
  "instance_type": {
    "id": "automatic"
  },
  "persistent_disk": {
    "size_gb": "20480"
  },
  "internet_connected": true,
  "elb_names": ["tcp:load_balancer_1", "http:load_balancer_2"],
  "additional_vm_extensions": ["vm_ext_setting_automatic_restart", "vm_ext_setting_on_host_maintenance"]
}'
```

For more information about this command, see Configuring resources for a job in the Ops Manager API docs.

See Deployed VM Extensions

From the `/api/v0/deployed/vm_extensions` endpoint, use `GET` to see your applied custom VM extensions.

See the following example command:

```
curl "https://example.com/api/v0/deployed/vm_extensions" \
-X GET \
-H "Authorization: Bearer UAA_ACCESS_TOKEN"
```

The output lists all your VM extension names and its associated custom properties defined in `cloud_properties`.

See the following example output:

```
{
  "vm_extensions": [
    {
      "name": "vm_ext1",
      "cloud_properties": {
        "source_dest_check": false
      }
    },
    {
      "name": "vm_ext2",
      "cloud_properties": {
        "key_name": "operations_keypair"
      }
    }
  ]
}
```

For more information about this command, see Creating a new VM extension in the Ops Manager API documentation.
Ops Manager Fields That Lock On Deploy

This topic describes how locked fields work in Ops Manager, and lists the unlockable and permanently-locked fields in installation tiles.

Overview

Some fields in Ops Manager lock during deployment. After deployment, a locked field is no longer configurable via the Ops Manager UI or API endpoints. Product tile authors configure fields as locked to prevent irrevocable VM malfunction, deployment corruption, and data loss. Most users will not need to edit locked fields after initial deployment.

Because deployment requirements may evolve in unanticipated ways over time, some locked fields are unlockable, which means that an advanced PCF operator can unlock them to allow reconfiguration. Other fields are permanently locked, and their values cannot be changed. Unlocking the unlockable fields requires you to use Ops Manager’s Advanced Mode. For more information about Advanced Mode, see Unlockable Fields.

In summary, there are three types of configurable fields in Ops Manager:

- **Standard fields**, which can be reconfigured after deployment as many times as you need.
- **Unlockable fields**, which lock on deployment but can be unlocked and configured with Advanced Mode.
- **Permanently-locked fields**, which lock on deployment and cannot be unlocked.

⚠️ warning: Pivotal recommends that only skilled operators use Ops Manager’s Advanced Mode. If you use Advanced Mode incorrectly, you could disable or destroy your deployment.

Unlockable Fields

This section describes the fields that lock during deployment and can be unlocked and configured after deployment. To unlock these fields, you must use Ops Manager’s Advanced Mode.

After you have enabled Advanced Mode, you can use the API or Ops Manager UI to reconfigure some fields that were locked previously. Those fields are listed in the following section.

⚠️ warning: Pivotal recommends that only skilled operators use Ops Manager’s Advanced Mode. If you use Advanced Mode incorrectly, you could disable or destroy your deployment.

Unlockable Fields in the Director Tile

In the Director Config pane, you can unlock the following fields after deployment:

- In the CredHub Encryption Provider fields, when you are using a Luna HSM, you can unlock the following fields after deployment:
  - Encryption Key Name
  - Provider Partition
  - HSM Host Address
  - HSM Port Address
  - Partition Serial Number

In the Assign AZs and Networks pane, you can unlock the following fields after deployment:

- The Singleton Availability Zone dropdown

Unlockable Fields in Product Tiles

Product tiles also have unlockable fields.

In the Assign AZs and Networks pane, you can unlock the following fields after deployment:

© Copyright Pivotal Software Inc, 2013-2019
- Place singleton jobs in
- Balance other jobs

Permanently-Locked Fields

This section describes the fields that lock during deployment and cannot be unlocked. These fields cannot be configured after deployment, either via the Ops Manager UI or via API endpoints.

Permanently-Locked Fields in the Director tile

In the Director Config pane, the following fields lock permanently after deployment:

In the Blobstore Location section:
- Enable TLS in the Internal section
- S3 Endpoint in the S3 Compatible Blobstore section
- Bucket Name in the S3 Compatible Blobstore section
- V2 Signature in the S3 Compatible Blobstore section
- V4 Signature in the S3 Compatible Blobstore section
- Region in the S3 Compatible Blobstore > V4 Signature section

Note: Changes to the Internal or S3 Compatible Blobstore radio buttons in the Blobstore Location section will not affect a deployed Ops Manager.

In the Database Location section:
- Host in the External MySQL Database section
- Port in the External MySQL Database section
- Username in the External MySQL Database section
- Password in the External MySQL Database section
- Database in the External MySQL Database section

Note: Changes to the Internal or External MySQL Database radio buttons in the Database Location section will not affect a deployed Ops Manager.

In the Create Availability Zone pane, the following fields lock permanently after deployment:
- You cannot delete an in-use Availability Zone (AZ)

In the Assign AZs and Networks pane, all fields lock permanently after deployment.

Configurable Fields in Product Tiles

No fields other than the ones listed above lock in third-party product tiles. However, tile authors can specify which additional fields should lock on deploy by passing the freeze-on-deploy flag.

For more information about this flag, see Common Property Blueprint Attributes.
PAS Concepts

Pivotal Application Service (PAS) is based on Cloud Foundry, which is an open source cloud application platform, providing a choice of clouds, developer frameworks, and application services. Cloud Foundry makes it faster and easier to build, test, deploy, and scale applications. It is an open source project and is available through a variety of private cloud distributions and public cloud instances.

This guide presents an overview of how Cloud Foundry works and a discussion of key concepts. Refer to this guide to learn more about Cloud Foundry fundamentals.

General Concepts

- Cloud Foundry Overview
- App Container Lifecycle
- How Apps Are Staged
- High Availability in Cloud Foundry
- Orgs, Spaces, Roles, and Permissions
- Application Security Groups
- Cloud Foundry Security
- Container Security
- Container-to-Container Networking
- GrootFS Disk Usage

Architecture

- Cloud Foundry Components
- Cloud Controller
- Messaging (NATS)
- Gorouter
- User Account and Authentication (UAA) Server
- Garden
- HTTP Routing

Diego

- Diego Architecture
- Application SSH Components and Processes
- How the Diego Auction Allocates Jobs
Cloud Foundry Overview

The Industry-Standard Cloud Platform

Cloud platforms let anyone deploy network apps or services and make them available to the world in a few minutes. When an app becomes popular, the cloud easily scales it to handle more traffic, replacing with a few keystrokes the build-out and migration efforts that once took months. Cloud platforms represent the next step in the evolution of IT, enabling you to focus exclusively on your applications and data without worrying about underlying infrastructure.

Not all cloud platforms are created equal. Some have limited language and framework support, lack key app services, or restrict deployment to a single cloud. Cloud Foundry (CF) has become the industry standard. It is an open source platform that you can deploy to run your apps on your own computing infrastructure, or deploy on an IaaS like AWS, vSphere, or OpenStack. You can also use a PaaS deployed by a commercial CF cloud provider. A broad community contributes to and supports Cloud Foundry. The platform’s openness and extensibility prevent its users from being locked into a single framework, set of app services, or cloud.

Cloud Foundry is ideal for anyone interested in removing the cost and complexity of configuring infrastructure for their apps. Developers can deploy their apps to Cloud Foundry using their existing tools and with zero modification to their code.

How Cloud Foundry Works

To flexibly serve and scale apps online, Cloud Foundry has subsystems that perform specialized functions. Here’s how some of these main subsystems work.

How the Cloud Balances Its Load

Clouds balance their processing loads over multiple machines, optimizing for efficiency and resilience against point failure. A Cloud Foundry installation accomplishes this at three levels:

1. BOSH creates and deploys virtual machines (VMs) on top of a physical computing infrastructure, and deploys and runs Cloud Foundry on top of this cloud. To configure the deployment, BOSH follows a manifest document.

2. The CF Cloud Controller runs the apps and other processes on the cloud’s VMs, balancing demand and managing app lifecycles.

3. The router routes incoming traffic from the world to the VMs that are running the apps that the traffic demands, usually working with a customer-provided load balancer.

How Apps Run Anywhere

Cloud Foundry designates two types of VMs: the component VMs that constitute the platform’s infrastructure, and the host VMs that host apps for the
outside world. Within CF, the Diego system distributes the hosted app load over all of the host VMs, and keeps it running and balanced through demand surges, outages, or other changes. Diego accomplishes this through an auction algorithm.

To meet demand, multiple host VMs run duplicate instances of the same app. This means that apps must be portable. Cloud Foundry distributes app source code to VMs with everything the VMs need to compile and run the apps locally. This includes the OS stack that the app runs on and a buildpack containing all languages, libraries, and services that the app uses. Before sending an app to a VM, the Cloud Controller stages it for delivery by combining stack, buildpack, and source code into a droplet that the VM can unpack, compile, and run. For simple, standalone apps with no dynamic pointers, the droplet can contain a pre-compiled executable instead of source code, language, and libraries.

How CF Organizes Users and Workspaces

CF manages user accounts through two User Authentication and Authorization (UAA) servers, which support access control as OAuth2 services and can store user information internally, or connect to external user stores through LDAP or SAML.

One UAA server grants access to BOSH, and holds accounts for the CF operators who deploy runtimes, services, and other software onto the BOSH layer directly. The other UAA server controls access to the Cloud Controller, and determines who can tell it to do what. The Cloud Controller UAA defines different user roles, such as admin, developer, or auditor, and grants them different sets of privileges to run CF commands. The Cloud Controller UAA also scopes the roles to separate, compartmentalized Orgs and Spaces within an installation, to manage and track use.

Where CF Stores Resources

Cloud Foundry uses the git system on GitHub to version-control source code, buildpacks, documentation, and other resources. Developers on the platform also use GitHub for their own apps, custom configurations, and other resources. To store large binary files, such as droplets, CF maintains an internal or external blobstore. CF uses MySQL to store and share temporary information, such as internal component states.

How CF Components Communicate

Cloud Foundry components communicate in two of the following ways:

- By sending messages internally using HTTP and HTTPS protocols
- By sending NATS messages to each other directly

BOSH Director collocates a BOSH DNS server on every deployed VM. All VMs keep up-to-date DNS records for all the other VMs in the same foundation. This enables service discovery between VMs.

BOSH DNS allows deployments to continue communicating with VMs even when the VMs' IP addresses change. It also provides client-side load-balancing by randomly selecting a healthy VM when multiple VMs are available.

For more information about BOSH DNS, see Native DNS Support in the BOSH documentation.

How to Monitor and Analyze a CF Deployment

Cloud Foundry generates system logs from Cloud Foundry components and app logs from hosted apps

As Cloud Foundry runs, its component and host VMs generate logs and metrics. Cloud Foundry apps also typically generate logs. The Loggregator system aggregates the component metrics and app logs into a structured, usable form, the Firehose. You can use all of the output of the Firehose, or direct the output to specific uses, such as monitoring system internals, triggering alerts, or analyzing user behavior, by applying nozzles.

The component logs follow a different path. They stream from rsyslog agents, and the cloud operator can configure them to stream out to a syslog drain.

Using Services with CF

Typical apps depend on free or metered services such as databases or third-party APIs. To incorporate these into an app, a developer writes a Service Broker, an API that publishes to the Cloud Controller the ability to list service offerings, provision the service, and enable apps to make calls out to it.

How Pivotal Cloud Foundry Differs from Open Source Cloud Foundry
Open source software provides the basis for the Pivotal Cloud Foundry platform. Pivotal Application Service (PAS) is the Pivotal distribution of Cloud Foundry software for hosting apps. Pivotal offers additional commercial features, enterprise services, support, documentation, certificates, and others value-adds.
How Apps Are Staged

This topic describes how Diego stages buildpack apps and Docker images.

Overview

Cloud Foundry uses Diego to manage app containers. It is a self-healing system that attempts to keep the correct number of instances running in Diego Cells to avoid network failures and crashes. For more information about Diego, see Diego Components and Architecture.

This topic references tasks and long-running processes (LRPs). For more information about these, see Tasks and Long-Running Processes.

How Diego Stages Buildpack Apps

This section describes how Diego stages buildpack apps.

The following diagram illustrates the steps and components involved in the process of staging a buildpack app.

1. A developer runs `cf push`.

2. The Cloud Foundry Command Line Interface (cf CLI) tells the Cloud Controller to create a record for the app. See the Cloud Controller topic for more information about the Cloud Controller.

3. The Cloud Controller stores the app metadata. App metadata can include the app name, number of instances, buildpack, and other information.
about the app.

4. This step includes the following:
   a. The cf CLI requests a resource match from the Cloud Controller.
   b. The cf CLI uploads the app source files, omitting any app files that already exist in the resource cache.
   c. The Cloud Controller combines the uploaded app files with files from the resource cache to create the app package.

5. The Cloud Controller stores the app package in the blobstore. For more information, see the Blobstore section in the Cloud Foundry Components topic.

6. The cf CLI issues a request to start the app.

7. This step includes the following:
   a. The Cloud Controller issues a staging request to Diego.
   b. Diego schedules a Diego Cell to run the staging task.
   c. The task downloads buildpacks and the app buildpack cache, if present.
   d. The task uses the buildpack to compile and stage the app.

8. The Diego Cell streams the output of the staging process. A developer may need to view the output to troubleshoot staging problems.

9. This step includes the following:
   a. The task creates a tarball, or droplet, with the compiled and staged app.
   b. The Diego Cell stores the droplet in the blobstore.
   c. The task uploads the buildpack cache to the blobstore for use the next time the app is staged.

10. The Diego Bulletin Board System (BBS) reports to the Cloud Controller that staging is complete. If staging does not complete within 15 minutes, it fails.

11. Diego schedules the app as a LRP on one or more Diego Cells.

12. The Diego Cells report the status of the app to the Cloud Controller.

How Diego Stages Docker Images

This section describes how Diego stages Docker images.

The following diagram illustrates the steps and components involved in the process of staging a Docker image.
For a description of each step in the process of staging a Docker image, see the following:

1. A developer runs `cf push` and includes the name of a Docker image in an accessible Docker Registry.

2. The cf CLI tells the Cloud Controller to create a record for the Docker image.

3. This step includes the following:
   a. The Cloud Controller issues a staging request to Diego.
   b. Diego schedules a Diego Cell to run the task.

4. The Diego Cell streams the output of the staging process. A developer may need to view the output to troubleshoot staging problems.

5. The task fetches the metadata associated with the Docker image and returns a portion of it to the Cloud Controller.

6. The Cloud Controller stores the metadata in the Cloud Controller database (CCDB).

7. This step includes the following:
   a. The Cloud Controller uses the Docker image metadata to construct a LRP that runs the start command specified in the Dockerfile.
   b. The Cloud Controller submits the LRP to Diego.
   c. Diego schedules the LRP on one or more Diego Cells.
   d. The Cloud Controller instructs Diego and the Gorouter to route traffic to the Docker image.

Note: The Cloud Controller takes into account any user-specified overrides specified in the Dockerfile, such as environment variables.
App Container Lifecycle

This topic describes the lifecycle of an app container for Cloud Foundry (CF) deployments running on the Diego architecture.

Deployment

The app deployment process involves uploading, staging, and starting the app in a container. Your app must successfully complete each of these phases within certain time limits. The default time limits for the phases are as follows:

- Upload: 15 minutes
- Stage: 15 minutes
- Start: 60 seconds

Note: Your administrator can change these defaults. Check with your administrator for the actual time limits set for app deployment.

Developers can change the time limit for starting apps through an app manifest or on the command line. For more information, see the timeout attribute section of the Deploying with App Manifests topic and Using Application Health Checks.

Crash Events

If an app instance crashes, CF automatically restarts it by rescheduling the instance on another container three times. After three failed restarts, CF waits thirty seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, CF stops trying to restart the app instance.

Evacuation

Certain operator actions require restarting VMs with containers hosting app instances. For example, an operator who updates stemcells or installs a new version of CF must restart all the VMs in a deployment.

CF automatically relocates the instances on VMs that are shutting down through a process called evacuation. CF recreates the app instances on another VM, waits until they are healthy, and then shuts down the old instances. During an evacuation, developers may see their app instances in a duplicated state for a brief period.

During this app duplication process, singleton app instances may become temporarily unavailable if the replacement instance does not become healthy within the cell's evacuation timeout, which defaults to 10 minutes. Because of this, app developers with a low tolerance for brief downtime may prefer to run several instances of their app. See Run Multiple Instances to Increase Availability.

Shutdown

PCF requests a shutdown of your app instance in the following scenarios:

- When a user runs cf scale, cf stop, cf push, cf delete, or cf restart-app-instance
- As a result of a system event, such as the replacement procedure during Diego cell evacuation or when an app instance stops because of a failed health-check probe

To shut down the app, CF sends the app process in the container a SIGTERM. The process has ten seconds to shut down gracefully. If the process has not exited after ten seconds, CF sends a SIGKILL.

Apps must finish their in-flight jobs within ten seconds of receiving the SIGTERM before CF terminates the app with a SIGKILL. For instance, a web app must finish processing existing requests and stop accepting new requests.

Note: One exception to the cases mentioned above is when monit restarts a crashed Diego cell rep or Garden server. In this case, CF immediately stops the apps that are still running using SIGKILL.
High Availability in Cloud Foundry

Components of a High Availability Deployment

Availability Zones

During product updates and platform upgrades, the VMs in a deployment restart in succession, rendering them temporarily unavailable. During outages, VMs go down in a less orderly way. Spreading components across Availability Zones (AZs) and scaling them to a sufficient level of redundancy maintains high availability during both upgrades and outages and can ensure zero downtime.

Deploying Cloud Foundry across three or more AZs and assigning multiple component instances to different AZ locations lets a deployment operate uninterrupted when entire AZs become unavailable. Cloud Foundry maintains its availability as long as a majority of the AZs remain accessible. For example, a three-AZ deployment stays up when one entire AZ goes down, and a five-AZ deployment can withstand an outage of up to two AZs with no impact on uptime.

External Load Balancers

Production environments should use a highly-available customer-provided load balancing solution that does the following:

- Provides load balancing to each of the Cloud Foundry Router IP addresses
- Supports SSL termination with wildcard DNS location
- Adds appropriate x-forwarded-for and x-forwarded-proto HTTP headers to incoming requests
- (Optional) Supports WebSockets

If you are deploying in lab and test environments, the `use-haproxy.yml` ops file enables HAProxy for your foundation.

For more information, see Using Your Own Load Balancer.

Blob Storage

For storing blobs, large binary files, the best approach for high availability is to use external storage such as Amazon S3 or an S3-compatible service.

If you store blobs internally using WebDAV or NFS, these components run as single instances and you cannot scale them. For these deployments, use the high availability features of your IaaS to immediately recover your WebDAV or NFS server VM if it fails. Contact Pivotal Support if you need assistance.

The singleton compilation components do not affect platform availability.

Vertical and Horizontal Scaling for High Availability

You can scale platform capacity vertically by adding memory and disk, or horizontally by adding more VMs running instances of Cloud Foundry components. The nature of the applications you host on Cloud Foundry should determine whether you should scale vertically or horizontally.
For more information about scaling applications and maintaining app uptime, see Scaling an Application Using cf scale and Using Blue-Green Deployment to Reduce Downtime and Risk.

Scale Vertically

Scaling vertically means adding memory and disk to your component VMs.

To scale vertically, ensure that you allocate and maintain enough of the following:

- Free space on host Diego cell VMs so that apps expected to deploy can successfully be staged and run.
- Disk space and memory in your deployment such that if one host VM is down, all instances of apps can be placed on the remaining Host VMs.
- Free space to handle one AZ going down if deploying in multiple AZs.

Scale Horizontally

Scaling horizontally means increasing the number of VM instances dedicated to running a functional component of the system.

You can horizontally scale most Cloud Foundry components to multiple instances to achieve the redundancy required for high availability.

You should also distribute the instances of multiply-scaled components across different AZs to minimize downtime during ongoing operation, product updates, and platform upgrades. If you use more than three AZs, ensure that you use an odd number of AZs.

For more information regarding zero downtime deployment, see Scaling Instances in PAS.

The following table provides recommended instance counts for a high-availability deployment and the minimum instances for a functional deployment:

<table>
<thead>
<tr>
<th>Pivotal Application Service (PAS) Job</th>
<th>Recommended Instance Number for HA</th>
<th>Minimum Instance Number</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diego Cell</td>
<td>≥ 3</td>
<td>1</td>
<td>The optimal balance between CPU/memory sizing and instance count depends on the performance characteristics of the apps that run on Diego cells. Scaling vertically with larger Diego cells makes for larger points of failure, and more apps go down when a cell fails. On the other hand, scaling horizontally decreases the speed at which the system re-balances apps. Re-balancing 100 cells takes longer and demands more processing overhead than re-balancing 20 cells.</td>
</tr>
<tr>
<td>Diego Brain</td>
<td>≥ 2</td>
<td>1</td>
<td>For high availability, use at least one per AZ, or at least two if only one AZ.</td>
</tr>
<tr>
<td>Diego BBS</td>
<td>≥ 2</td>
<td>1</td>
<td>For high availability in a multi-AZ deployment, use at least one instance per AZ. Scale Diego BBS to at least two instances for high availability in a single-AZ deployment.</td>
</tr>
<tr>
<td>MySQL Server</td>
<td>3</td>
<td>1</td>
<td>If you use an external database in your deployment, then you can set the MySQL Server instance count to 0. For instructions about scaling down an internal MySQL cluster, see Scaling Down Your MySQL Cluster.</td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>2</td>
<td>1</td>
<td>If you use an external database in your deployment, then you can set the MySQL Proxy instance count to 0. In a high availability deployment, you might run a single NATS instance if your deployment lacks the</td>
</tr>
</tbody>
</table>
Configure Support for High Availability Components

This section describes the surrounding infrastructure required to support scaling component VMs for high availability.

BOSH Resurrector

The BOSH Resurrector increases Pivotal Application Service (PAS) availability in the following ways:

- Reacts to hardware failure and network disruptions by recreating virtual machines on active, stable hosts
- Detects operating system failures by continuously monitoring virtual machines and recreating them as required
- Continuously monitors the BOSH Agent running on each virtual machine and recreates the VMs as required

The BOSH Resurrector continuously monitors the status of all virtual machines in a PAS deployment. The Resurrector also monitors the BOSH Agent on each VM. If either the VM or the BOSH Agent fail, the Resurrector recreates the virtual machine on another active host. To enable the BOSH Resurrector, see Enable BOSH Resurrector.

Resource Pools

To configure your resource pools according to the requirements of your deployment, see the Ops Manager configuration topic for your IaaS.

Each IaaS has different ways of limiting resource consumption for scaling VMs. Consult with your IaaS administrator to ensure additional VMs and related resources, like IPs and storage, will be available when scaling.

For information about configuring resource pools for Amazon Web Services, see Amazon EC2 FAQs in the Amazon documentation. For information about configuring resource pools for OpenStack, see Manage projects and users in the OpenStack documentation. For information about configuring resource pools for vSphere, see Configuring BOSH Director on vSphere.

Databases

For database services deployed outside Cloud Foundry, plan to leverage your infrastructure’s high availability features and to configure backup and restore where possible. For more information about scaling internal database components, see the Scaling Instances in PAS topic.

Note: Data services may have single points of failure depending on their configuration.

Contact Pivotal Support if you need assistance.
How Cloud Foundry Maintains High Availability

Page last updated:

This topic explains how Pivotal Cloud Foundry (PCF) deployments include several layers of high availability to keep applications running during system failure. These layers include AZs, application health management, process monitoring, and VM resurrection.

Availability Zones

PCF supports deploying applications instances across multiple AZs. This level of high availability requires that you define AZs in your IaaS. PCF balances the applications you deploy across the AZs you defined. If an AZ goes down, you still have application instances running in another.

You can configure your deployment so that Diego cells are created across these AZs. See the Assign AZs and Networks pane of the PAS tile.

Health Management for App Instances

If you lose application instances for any reason, such as a bug in the app or an AZ going down, PCF restarts new instances to maintain capacity. Under Diego architecture, the nsync, BBS, and Cell Rep components track the number of instances of each application that are running across all of the Diego cells. When these components detect a discrepancy between the actual state of the app instances in the cloud and the desired state as known by the Cloud Controller, they advise the Cloud Controller of the difference and the Cloud Controller initiates the deployment of new application instances.

For more information about the nsync, BBS, and Cell Rep components, see the nsync, BBS, and Cell Rep section of the Cloud Foundry Components topic.

Process Monitoring

PCF uses a BOSH agent, monit, to monitor the processes on the component VMs that work together to keep your applications running, such as nsync, BBS, and Cell Rep. If monit detects a failure, it restarts the process and notifies the BOSH agent on the VM. The BOSH agent notifies the BOSH Health Monitor, which triggers responders through plugins such as email notifications or paging.

Resurrection for VMs

BOSH detects if a VM is present by listening for heartbeat messages that are sent from the BOSH agent every 60 seconds. The BOSH Health Monitor listens for those heartbeats. When the Health Monitor finds that a VM is not responding, it passes an alert to the Resurrector component. If the Resurrector is enabled, it sends the IaaS a request to create a new VM instance to replace the one that failed.

To enable the Resurrector, see the following pages for your particular IaaS: AWS, Azure, GCP, OpenStack, or vSphere.
Orgs, Spaces, Roles, and Permissions

PCF uses a role-based access control (RBAC) system to grant Pivotal Application Service (PAS) users permissions appropriate to their role within an org or a space. This topic describes how orgs and spaces work within a PCF deployment, and how different Pivotal Application Service (PAS) User roles operate within those contexts.

Admins, Org Managers, and Space Managers can assign user roles using the cf CLI or Apps Manager.

Note: Before you assign a space role to a user, you must assign an org role to the user.

Orgs

An org is a development account that an individual or multiple collaborators can own and use. All collaborators access an org with user accounts. Collaborators in an org share a resource quota plan, applications, services availability, and custom domains.

By default, an org has the status of active. An admin can set the status of an org to suspended for various reasons such as failure to provide payment or misuse. When an org is suspended, users cannot perform certain activities within the org, such as push apps, modify spaces, or bind services. For details on what activities are allowed for suspended orgs, see Roles and Permissions for Suspended Orgs.

User Accounts

A user account represents an individual person within the context of a PCF installation. A user can have different roles in different spaces within an org, governing what level and type of access they have within that space.

Before you assign a space role to a user, you must assign an org role to the user. The error message

Server error, error code: 1002, message: cannot set space role because user is not part of the org

occurs when you try to set a space role before setting an org role for the user.

Spaces

Every application and service is scoped to a space. An org can contain multiple spaces. A space provides users with access to a shared location for application development, deployment, and maintenance. Each space role applies only to a particular space.

Roles and Permissions

A user can have one or more roles. The combination of these roles defines the user’s overall permissions in the org and within specific spaces in that org. Roles can be assigned different scopes of User Account and Authentication (UAA) privileges. For more information about UAA scopes, see Scopes in Component: User Account and Authentication (UAA) Server.

For non-admin users, the cloud_controller.read scope is required to view resources, and the cloud_controller.write scope is required to create, update, and delete resources.

- Admin is a user role that has been assigned the cloud_controller.admin scope in UAA. An admin user has permissions on all orgs and spaces and can perform operational actions using the Cloud Controller API. To create an account with cloud_controller.admin scope for your installation, see Create an Admin User.

- Admin Read-Only is a user role that has been assigned the cloud_controller.admin_read_only scope in UAA. This role has read-only access to all Cloud Controller API resources.

- Global Auditor is a user role that has been assigned the cloud_controller.global_auditor scope in UAA. This role has read-only access to all Cloud Controller API resources except for secrets such as environment variables. The Global Auditor role cannot access those values.

- Org Managers are managers or other users who need to administer the org.

- Org Auditors view but cannot edit user information and org quota usage information.

- Org Users can view the list of other org users and their roles. When an Org Manager gives a person an Org or Space role, that person automatically receives Org User status in that Org.

- Space Managers are managers or other users who administer a space within an org.
Space Developers are application developers or other users who manage applications and services in a space.

Space Auditors view but cannot edit the space.

Roles and Permissions for Active Orgs

The following table describes the permissions for various PCF roles.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Admin Read-Only</th>
<th>Global Auditor</th>
<th>Org Manager</th>
<th>Org Auditor</th>
<th>Space Manager</th>
<th>Space Developer</th>
<th>Space Auditor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope of operation</td>
<td>Org</td>
<td>Org</td>
<td>Org</td>
<td>Org</td>
<td>Space</td>
<td>Space</td>
<td>Space</td>
</tr>
<tr>
<td>Add and edit users and roles</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create and assign org quota plans</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create orgs</td>
<td>✓</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>View all orgs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View orgs where user is a member</td>
<td>✓³</td>
<td>✓³</td>
<td>✓³</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Edit, rename, and delete orgs</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓⁴</td>
</tr>
<tr>
<td>Suspend or activate an org</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create and assign space quota plans</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create spaces</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>View spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Edit spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Delete spaces</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rename spaces</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>View the status, number of instances, service bindings, and resource use of applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Add private domains³</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Deploy, run, and manage applications</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Use application SSH⁴</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Instantiate and bind services to applications</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associate routes³, instance counts, memory allocation, and disk limit of applications</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Rename applications</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Create and manage Application Security Groups</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create, update, and delete an Isolation Segment</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List all Isolation Segments for an Org</td>
<td>✓</td>
<td>✓</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓¹</td>
</tr>
<tr>
<td>List all Orgs entitled to an Isolation Segment</td>
<td>✓</td>
<td>✓</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓¹</td>
</tr>
<tr>
<td>Assign a default Isolation Segment to an Org</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>List and manage Isolation Segments for spaces</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>List entitled Isolation Segment for a space</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>List which Isolation Segment an app runs on</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

¹Not by default, unless feature flag set_roles_by_username is set to true.

²Not by default, unless feature flag user_org_creation is set to true.

³Admin, admin read-only, and global auditor roles do not need to be added as members of orgs or spaces to view resources.

⁴Org Managers can rename their orgs and edit some fields; they cannot delete orgs.
Roles and Permissions for Suspended Orgs

The following table describes roles and permissions applied after an operator sets the status of an org to `suspended`.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Admin</th>
<th>Org</th>
<th>Org</th>
<th>Org</th>
<th>Org</th>
<th>Space</th>
<th>Space</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope of operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add and edit users and roles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View users and roles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create and assign org quota plans</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View org quota plans</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create orgs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View all orgs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View orgs where user is a member</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Edit, rename, and delete orgs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Suspend or activate an org</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create and assign space quota plans</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Edit spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Delete spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rename spaces</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>View the status, number of instances, service bindings, and resource use of applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Add private domains&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Deploy, run, and manage applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Instantiate and bind services to applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Associate routes&lt;sup&gt;1&lt;/sup&gt;, instance counts, memory allocation, and disk limit of applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rename applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create and manage Application Security Groups</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<sup>1</sup>Unless disabled by feature flags.

<sup>6</sup>This assumes that SSH is enabled for the platform, space, and app. For more information, see [SSH Access Control Hierarchy](#).

<sup>7</sup>Applies only to orgs they belong to.
Cloud Foundry Security

Page last updated:

This topic provides an overview of Cloud Foundry (CF) security. For an overview of container security, see the Container Security topic.

Cloud Foundry implements the following measures to mitigate against security threats:

- Minimizes network surface area
- Isolates customer applications and data in containers
- Encrypts connections
- Uses role-based access controls, applying and enforcing roles and permissions to ensure that users can only view and affect the spaces for which they have been granted access
- Ensures security of application bits in a multi-tenant environment
- Prevents possible denial of service attacks through resource starvation

System Boundaries and Access

As the image below shows, in a typical deployment of Cloud Foundry, the components run on virtual machines (VMs) that exist within a VLAN. In this configuration, the only access points visible on a public network are a load balancer that maps to one or more Cloud Foundry routers and, optionally, a NAT VM and a jumpbox. Because of the limited number of contact points with the public internet, the surface area for possible security vulnerabilities is minimized.

Note: Pivotal recommends that you also install a NAT VM for outbound requests and a Jumpbox to access the BOSH Director, though these access points are optional depending on your network configuration.
Protocols

All traffic from the public internet to the Cloud Controller and UAA happens over HTTPS. Inside the boundary of the system, components communicate over a publish-subscribe (pub-sub) message bus NATS, HTTP, and SSL/TLS.

BOSH

Operators deploy Cloud Foundry with BOSH. The BOSH Director is the core orchestrating component in BOSH: it controls VM creation and deployment, as well as other software and service lifecycle events. You use HTTPS to ensure secure communication to the BOSH Director.

Note: Pivotal recommends that you deploy the BOSH Director on a subnet that is not publicly accessible, and access the BOSH Director from a Jumpbox on the subnet or through VPN.

BOSH includes the following functionality for security:

- Communicates with the VMs it launches over NATS. Because NATS cannot be accessed from outside Cloud Foundry, this ensures that published messages can only originate from a component within your deployment.
- Provides an audit trail through the `bosh tasks --all` and `bosh tasks --recent=VALUE` commands, `bosh tasks --all` returns a table that shows all BOSH actions taken by an operator or other running processes, `bosh tasks --recent=VALUE` returns a table of recent tasks, with `VALUE` being the number of recent tasks you want to view.
- Allows you to set up individual login accounts for each operator. BOSH operators have root access.

Note: BOSH does not encrypt data stored on BOSH VMs. Your IaaS might encrypt this data.
Isolation Segments

Isolation segments provide dedicated pools of resources to which apps can be deployed to isolate workloads. Using isolation segments separates app resources as completely as if they were in different CF deployments but avoids redundant management components and unneeded network complexity.

You can designate isolation segments for exclusive use by orgs and spaces within CF. This guarantees that apps within the org or space use resources that are not also used by other orgs or spaces.

Customers can use isolation segments for different reasons, including the following:

- To follow regulatory restrictions that require separation between different types of applications. For example, a health care company may not be able to host medical records and billing systems on the same machines.
- To dedicate specific hardware to different isolation segments. For example, to guarantee that high-priority apps run on a cluster of high-performance hosts.
- To separate data on multiple clients, to strengthen a security story, or offer different hosting tiers.

In CF, the Cloud Controller Database (CCDB) identifies isolation segments by name and GUID, for example: 30dd879c-ee2f-11db-8314-0800200c9a66. The isolation segment object has no internal structure beyond these two properties at the Cloud Foundry level, but BOSH associates the name of the isolation segment with Diego cells, through their placement_tag property.

This diagram shows how isolation segments keep apps running on different pools of cells, and how the cells communicate with each other and with the management components:

![Isolation Segment Diagram]

See the Installing PCF Isolation Segment and Managing Isolation Segments topics for more information about how to create and manage isolation segments in a PCF deployment.

See the Isolation Segments section of the Cloud Controller API (CAPI) Reference for API commands related to isolation segments.

Authentication and Authorization

User Account and Authentication (UAA) is the central identity management service for Cloud Foundry and its various components.

UAA acts as an OAuth2 Authorization Server and issues access tokens for applications that request platform resources. The tokens are based on the JSON Web Token and are digitally signed by UAA.

Operators can configure the identity store in UAA. If users register an account with the Cloud Foundry platform, UAA acts as the user store and stores user passwords in the UAA database using bcrypt. UAA also supports connecting to external user stores through LDAP and SAML. Once an operator has
configured the external user store, such as a corporate Microsoft Active Directory, users can use their LDAP credentials to gain access to the Cloud Foundry platform instead of registering a separate account. Alternatively, operators can use SAML to connect to an external user store and enable single sign-on for users into the Cloud Foundry platform.

Managing User Access with Role-Based Access Control

Applications that users deploy to Cloud Foundry exist within a space. Spaces exist within orgs. To view and access an org or a space, a user must be a member of it. Cloud Foundry uses role-based access control (RBAC), with each role granted permissions to either an org or a specified space. For more information about roles and permissions, refer to the Orgs, Spaces, Roles, and Permissions topic.

For more information, see Getting Started with the Apps Manager and Managing User Accounts and Permissions Using the Apps Manager.

Security for Service Broker Integration

The Cloud Controller authenticates every request with the Service Broker API using HTTP or HTTPS, depending on which protocol that you specify during broker registration. The Cloud Controller rejects any broker registration that does not contain a username and password.

Service instances bound to an app contain credential data. Users specify the binding credentials for user-provided service instances, while third-party brokers specify the binding credentials for managed service instances. The VCAP_SERVICES environment variable contains credential information for any service bound to an app. Cloud Foundry constructs this value from encrypted data that it stores in the Cloud Controller Database (CCDB).

Note: The selected third-party broker controls how securely to communicate managed service credentials.

A third-party broker might offer a dashboard client in its catalog. Dashboard clients require a text string defined as a client_secret. Cloud Foundry does not store this secret in the CCDB. Instead, Cloud Foundry passes the secret to the UAA component for verification using HTTP or HTTPS.

Software Vulnerability Management

Cloud Foundry manages software vulnerability using releases and BOSH stemcells. New Cloud Foundry releases are created with updates to address code issues, while new stemcells are created with patches for the latest security fixes to address any underlying operating system issues.

Ensuring Security for Application Artifacts

Cloud Foundry secures both the code and the configuration of an application using the following functionality:

- Application developers push their code using the Cloud Foundry API. Cloud Foundry secures each call to the CF API using the UAA and SSL.
- The Cloud Controller uses RBAC to ensure that only authorized users can access a particular application.
- The Cloud Controller stores the configuration for an application in an encrypted database table. This configuration data includes user-specified environment variables and service credentials for any services bound to the app.
- Cloud Foundry runs the app inside a secure container. For more information, see the Container Security topic.
- Cloud Foundry operators can configure network traffic rules to control inbound communication to and outbound communication from an app. For more information, see the Network Traffic Rules section of the Container Security topic.

Security Event Logging and Auditing

For operators, Cloud Foundry provides an audit trail through the bosh tasks command. This command shows all actions that an operator has taken with the platform. Additionally, operators can redirect Cloud Foundry component logs to a standard syslog server using the syslog_daemon_config property in the metron_agent job of cf-release.

For users, Cloud Foundry records an audit trail of all relevant API invocations of an app. The Cloud Foundry Command Line Interface (cf CLI) command returns this information.
Recommendations for Running a Secure Deployment

To help run a secure deployment, Pivotal recommends the following:

- Configure UAA clients and users using a BOSH manifest. Limit and manage these clients and users as you would any other kind of privileged account.
- Deploy within a VLAN that limits network traffic to individual VMs. This reduces the possibility of unauthorized access to the VMs within your BOSH-managed cloud.
- Enable HTTPS for applications and SSL database connections to protect sensitive data transmitted to and from applications.
- Ensure that the Jumpbox is secure, along with the load balancer and NAT VM.
- Encrypt stored files and data within databases to meet your data security requirements. Deploy using industry standard encryption and the best practices for your language or framework.
- Prohibit promiscuous network interfaces on the trusted network.
- Review and monitor data sharing and security practices with third-party services that you use to provide additional functionality to your application.
- Store SSH keys securely to prevent disclosure, and promptly replace lost or compromised keys.
- Use Cloud Foundry’s RBAC model to restrict your users’ access to only what is necessary to complete their tasks.
- Use a strong passphrase for both your Cloud Foundry user account and SSH keys.
- Use the IPsec add-on to encrypt IP data traffic within your deployment.
Container Security

This topic describes how Cloud Foundry (CF) secures the containers that host application instances on Linux. For an overview of other CF security features, see the Cloud Foundry Security topic.

- Container Mechanics provides an overview of container isolation.
- Inbound and Outbound Traffic from CF provides an overview of container networking and describes how CF administrators customize container network traffic rules for their deployment.
- Container Security describes how CF secures containers by running application instances in unprivileged containers and by hardening them.

Container Mechanics

Each instance of an app deployed to CF runs within its own self-contained environment, a Garden container. This container isolates processes, memory, and the filesystem using operating system features and the characteristics of the virtual and physical infrastructure where CF is deployed.

CF achieves container isolation by namespacing kernel resources that would otherwise be shared. The intended level of isolation is set to prevent multiple containers that are present on the same host from detecting each other. Every container includes a private root filesystem, which includes a Process ID (PID), namespace, network namespace, and mount namespace.

CF creates container filesystems using the Garden Rootfs (GrootFS) tool. It stacks the following using OverlayFS:

- A read-only base filesystem: This filesystem has the minimal set of operating system packages and Garden-specific modifications common to all containers. Containers can share the same read-only base filesystem because all writes are applied to the read-write layer.
- A container-specific read-write layer: This layer is unique to each container and its size is limited by XFS project quotas. The quotas prevent the read-write layer from overflowing into unallocated space.

Resource control is managed using Linux control groups. Associating each container with its own cgroup or job object limits the amount of memory that the container may use. Linux cgroups also require the container to use a fair share of CPU compared to the relative CPU share of other containers.

Note: CF does not support a RedHat Enterprise Linux OS stemcell. This is due to an inherent security issue with the way RedHat handles user namespacing and container isolation.

CPU

Each container is placed in its own cgroup. Cgroups make each container use a fair share of CPU relative to the other containers. This prevents oversubscription on the host VM where one or more containers hog the CPU and leave no computing resources to the others.

The way cgroups allocate CPU time is based on shares. CPU shares do not work as direct percentages of total CPU usage. Instead, a share is relative in a given time window to the shares held by the other containers. The total amount of CPU that can be overall divided among the cgroups is what is left by other processes that may run in the host VM.

Generally, cgroups offers two possibilities for limiting the CPU usage: CPU affinity and CPU bandwidth, the latter of which is used in Cloud Foundry.

- CPU affinity: It consists of binding a cgroup to certain CPU cores. The actual amount of CPU cycles that can be consumed by the cgroup is thus limited to what is available on the bound CPU cores.
- CPU bandwidth: Sets the weight of a cgroup with the process scheduler. The process scheduler divides the available CPU cycles among cgroups depending on the shares held by each cgroup, relative to the shares held by the others.

For example, consider two cgroups, one holding two shares and one holding four. Assuming the process scheduler gets to administer 60 CPU cycles, the first cgroup with two shares will get one third of those available CPU cycles, as it holds one third of the overall shares. Similarly, the second cgroup will get 40 cycles, as it holds two thirds of the collective shares.

The calculation of the CPU usage based on the percentage of the total CPU power available is quite sophisticated and is performed regularly as the CPU demands of the various containers fluctuates. Specifically, the percentage of CPU cycles a cgroup gets can be calculated by dividing the sum of the cgroup's shares by the total shares of all the cgroups that are currently doing CPU activity:

```
process_cpu_share_percent = cgroup.s shares / sum(cgroup.s shares) * 100
```

In Cloud Foundry, cgroup shares range from 10 to 1024, with 1024 being the default. The actual amount of shares a cgroup gets can be read from the
The amount of shares given to an application’s cgroup depends on the amount of memory the application declares to need in the manifest. Cloud Foundry scales the number of allocated shares linearly with the amount of memory, with an app instance requesting 8G of memory getting the upper limit of 1024 shares.

\[
\text{process.cpu.shares} = \min\left(1024\times\frac{\text{application_memory}}{8\text{ GB}}, 1024\right)
\]

The next example helps to illustrate this better. Consider three processes, P1, P2 and P3, which are assigned cpu.shares of 5, 20 and 30, respectively.

P1 is active, while P2 and P3 require no CPU. Hence, P1 may use the whole CPU. When P2 joins in and is doing some actual work (e.g. a request comes in), the CPU share between P1 and P2 will be calculated as follows:

- P1 = \(5 / (5+20) = 0.2 = 20\%\)
- P2 = \(20 / (5+20) = 0.8 = 80\%\)
- P3 (idle)

At some point process P3 joins in as well. Then the distribution will be recalculated again:

- P1 = \(5 / (5+20+30) = 0.0909 = \sim9\%\)
- P2 = \(20 / (5+20+30) = 0.363 = \sim36\%\)
- P3 = \(30 / (5+20+30) = 0.545 = \sim55\%\)

Should P1 become idle, the following recalculation between P2 and P3 takes place:

- P1 (idle)
- P2 = \(20 / (20+30) = 0.4 = 40\%\)
- P3 = \(30 / (20+30) = 0.6 = 60\%\)

If P3 finishes or becomes idle then P2 can consume all the CPU as another recalculation will be performed.

---

**Summary:** The cgroup shares are the minimum guaranteed CPU share that the process can get. This limitation becomes effective only when processes on the same host compete for resources.

---

### Inbound and Outbound Traffic from CF

#### Networking Overview

A host VM has a single IP address. If you configure the deployment with the cluster on a VLAN, as recommended, then all traffic goes through the following levels of network address translation, as shown in the diagram below.

- **Inbound** requests flow from the load balancer through the router to the host cell, then into the application container. The router determines which application instance receives each request.
- **Outbound** traffic flows from the application container to the cell, then to the gateway on the cell’s virtual network interface. Depending on your IaaS, this gateway may be a NAT to external networks.
Network Traffic Rules

Administrators configure rules to govern container network traffic. This is how containers send traffic outside of CF and receive traffic from outside, the Internet. These rules can prevent system access from external networks and between internal components and determine if apps can establish connections over the virtual network interface.

Administrators configure these rules at two levels:

- Application Security Groups (ASGs) apply network traffic rules at the container level. For information about creating and configuring ASGs, see Application Security Groups.
- Container-to-Container networking policies determine app-to-app communication. Within CF, apps can communicate directly with each other, but the containers are isolated from outside CF. For information about administering container-to-container network policies, see Administering Container-to-Container Networking.

Container Security

CF secures containers through the following measures:

- Running application instances in unprivileged containers by default
- Hardening containers by limiting functionality and access rights
- Allowing administrators to configure ASGs to block outbound connections from application containers. For information about creating and configuring ASGs, see Application Security Groups.

Types

Garden has two container types: unprivileged and privileged. Currently, CF runs all application instances and staging tasks in unprivileged containers by default. This measure increases security by eliminating the threat of root escalation inside the container.

Hardening

CF mitigates against container breakout and denial of service attacks in the following ways:

- CF uses the full set of Linux namespaces (IPC, Network, Mount, PID, User, UTS) to provide isolation between containers running on the same host. The User namespace is not used for privileged containers.
- In unprivileged containers, CF maps UID/GID 0 (root) inside the container user namespace to a different UID/GID on the host to prevent an app from inheriting UID/GID 0 on the host if it breaks out of the container.
  - CF uses the same UID/GID for all containers.
  - CF maps all UIDs except UID 0 to themselves. CF maps UID 0 inside the container namespace to MAX_UID-1 outside of the container namespace.
  - Container Root does not grant Host Root permissions.
- CF mounts /proc and /sys as read-only inside containers.
- CF disallows /dev/ access for unprivileged users and all users in unprivileged containers.
- CF uses chroot when importing docker images from docker registries.
- CF establishes a container-specific overlay filesystem mount. CF uses pivot_root to move the root filesystem into this overlay, in order to isolate the container from the host system’s filesystem.
- CF does not call any binary or script inside the container filesystem, in order to eliminate any dependencies on scripts and binaries inside the root filesystem.
- CF avoids side-loading binaries in the container through bind mounts or other methods. Instead, it re-executes the same binary by reading it from /proc/self/exe whenever it needs to run a binary in a container.
- CF establishes a virtual Ethernet pair for each container for network traffic. See the Container Network Traffic section above for more information. The virtual Ethernet pair has the following features:
  - One interface in the pair is inside the container’s network namespace, and is the only non-loopback interface accessible inside the container.
  - The other interface remains in the host network namespace and is bridged to the container-side interface.
  - Egress whitelist rules are applied to these interfaces according to Application Security Groups (ASGs) configured by the administrator.
  - First-packet logging rules may also be enabled on TCP whitelist rules.
  - DNAT rules are established on the host to enable traffic ingress from the host interface to whitelisted ports on the container-side interface.
- CF applies disk quotas using container-specific XFS quotas with the specified disk-quota capacity.
- CF applies a total memory usage quota through the memory cgroup and destroys the container if the memory usage exceeds the quota.
- CF applies a fair-use limit to CPU usage for processes inside the container through the `cpu.shares` control group.
- CF allows administrators to rate limit the maximum bandwidth consumed by single application containers, configuring `rate` and `burst` properties on the `silk-cni` job.
- CF limits access to devices using cgroups but explicitly whitelists the following safe device nodes:
  - `/dev/full`
  - `/dev/fuse`
  - `/dev/null`
  - `/dev/ptmx`
  - `/dev/pts/*`
  - `/dev/random`
  - `/dev/tty`
  - `/dev/tty0`
  - `/dev/tty1`
  - `/dev/urandom`
  - `/dev/zero`
  - `/dev/tap`
  - `/dev/tun`
- CF drops the following Linux capabilities for all container processes. Every dropped capability limits the actions the root user can perform.
  - `CAP_DAC_READ_SEARCH`
  - `CAP_LINUX_IMMUTABLE`
  - `CAP_NET_BROADCAST`
  - `CAP_NET_ADMIN`
  - `CAP_IPC_LOCK`
  - `CAP_IPC_OWNER`
  - `CAP_SYS_MODULE`
  - `CAP_SYS_RAWIO`
  - `CAP_IPC_OWNER`
  - `CAP_SYS_PACCT`
  - `CAP_SYS_BOOT`
  - `CAP_SYS_NICE`
  - `CAP_SYS_RESOURCE`
  - `CAP_SYS_TIME`
  - `CAP_SYS_TTY_CONFIG`
  - `CAP_LEASE`
  - `CAP_AUDIT_CONTROL`
  - `CAP_MAC_OVERRIDE`
  - `CAP_MAC_ADMIN`
  - `CAP_SYSLOG`
  - `CAP_WAKE_ALARM`
  - `CAP_BLOCK_SUSPEND`
  - `CAP_SYS_ADMIN` (for unprivileged containers)
Container-to-Container Networking

This topic provides an overview of how Container-to-Container Networking works.

The Container-to-Container Networking feature enables app instances to communicate with each other directly. Container-to-Container Networking is always enabled in PAS. For more information about how to configure Container-to-Container Networking, see the Administering Container-to-Container Networking topic.

Architecture

Overview

Container-to-Container Networking integrates with Garden-runC in a Diego deployment. The Container-to-Container Networking BOSH release includes several core components, as well as swappable components.

To understand the components and how they work, see the diagram and tables below. The diagram highlights Pivotal Application Service (PAS) components in blue and green. The diagram also highlights swappable components in red.

![Diagram of Container-to-Container Networking](image)

Core Components

The Container-to-Container Networking BOSH release includes the following core components:

<table>
<thead>
<tr>
<th>Part</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Foundry Command Line Interface (CF CLI) plugin</td>
<td>A plugin that you download to control network access policies between apps.</td>
</tr>
<tr>
<td></td>
<td>A central management node that does the following:</td>
</tr>
<tr>
<td></td>
<td>• Maintains a database of policies for traffic between apps. The CF CLI plugin calls an API to create or update a</td>
</tr>
</tbody>
</table>
Policy Server
- Exposes a JSON REST API used by the CF CLI plugin

Garden External Networker
- A Garden-run C add-on deployed to every Diego cell that does the following:
  - Invokes the CNI plugin component to set up the network for each app
  - Forwards ports to support incoming connections from the Gorouter, TCP Router, and Diego SSH Proxy. This keeps apps externally reachable.
  - Installs outbound whitelist rules to support Application Security Groups (ASGs).

### Swappable Components

The Container-to-Container Networking BOSH release includes the following swappable components:

<table>
<thead>
<tr>
<th>Part</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Silk CNI plugin</td>
<td>A plugin that provides IP address management and network connectivity to app instances as follows:</td>
</tr>
<tr>
<td></td>
<td>- Uses a shared VXLAN overlay network to assign each container a unique IP address</td>
</tr>
<tr>
<td></td>
<td>- Installs network interface in container using the Silk VXLAN backend. This is a shared, flat L3 network.</td>
</tr>
<tr>
<td>VXLAN Policy Agent</td>
<td>Enforces network policy for traffic between apps as follows:</td>
</tr>
<tr>
<td></td>
<td>- Discovers desired network policies from the Policy Server Internal API</td>
</tr>
<tr>
<td></td>
<td>- Updates iptables rules on the Diego cell to allow whitelisted inbound traffic</td>
</tr>
<tr>
<td></td>
<td>- Tags outbound traffic with the unique identifier of the source app using the VXLAN Group-Based Policy (GBP) header</td>
</tr>
</tbody>
</table>

### App Instance Communication

The diagram below illustrates how app instances communicate in a deployment with Container-to-Container Networking enabled. In this example, the operator creates two policies to regulate the flow of traffic between App A, App B, and App C.

- Allow traffic from App A to App B
- Allow traffic from App A to App C

If traffic and its direction is not explicitly allowed, it is denied. For example, App B cannot send traffic to App C.

#### Overlay Network

Container-to-Container Networking uses an overlay network to manage communication between app instances.
Overlay networks are not externally routable, and traffic sent between containers does not exit the overlay. You can use the same overlay network range for different Cloud Foundry deployments in your environment.

The overlay network range defaults to `/10.255.0.0/16`. You can modify the default to any RFC 1918 range that meets the following requirements:

- The range is not used by services that app containers access.
- The range is not used by the underlying Cloud Foundry infrastructure.

All Diego cells in your Cloud Foundry deployment share this overlay network. By default, each cell is allocated a /24 range that supports 254 containers per cell, one container for each of the usable IP addresses, 1 through 254. To modify the number of Diego cells your overlay network supports, see Overlay Network in Configuring Container-to-Container Networking.

Cloud Foundry container networking is currently supported only on Linux.

**Warning:** The overlay network IP address range must not conflict with any other IP addresses in the network. If a conflict exists, Diego cells cannot reach any endpoint that has a conflicting IP address.

**Note:** Traffic to app containers from the Gorouter or from app containers to external services uses cell IP addresses and NAT, not the overlay network.

### Policies

Enabling Container-to-Container Networking for your deployment allows you to create policies for communication between app instances. The Container-to-Container Networking feature also provides a unique IP address to each app container and provides direct IP reachability between app instances.

The policies you create specify a source app, destination app, protocol, and port so that app instances can communicate directly without going through the Gorouter, a load balancer, or a firewall. Container-to-Container Networking supports UDP and TCP, and you can configure policies for multiple ports. These policies apply immediately without having to restart the app.

Additionally, policies use and track the GUIDs of the apps. The policies continue to work when apps redeploy, or if they crash and Diego places them in a new container. Pushing a brand new app requires a new policy, but not updates to an existing app because an app always retains its GUID.

### App Service Discovery

The Pivotal Application Service (PAS) platform supports DNS-based service discovery that lets apps find each others' internal addresses. For example, a front end app instance can use the service discovery mechanism to establish communications with a back end app instance. See the Developer Guide for how to set up and use app service discovery.

Container-to-Container app service discovery does not provide client-side load balancing or circuit-breaking, and it does not apply to services or require application binding. It just lets apps publish service endpoints to each other, unbrokered and unmediated.

### Alternative Network Stacks

The BOSH release that contains the Container-to-Container Networking feature is composed of a pluggable network stack. Advanced users or third-party vendors can integrate a different network stack. For more information about third-party plugins, see the Container-to-Container Networking BOSH release documentation.

### Container-to-Container Networking versus ASGs

Both application security groups (ASGs) and Container-to-Container Networking policies affect traffic from app instances. The following table highlights differences between ASGs and Container-to-Container Networking policies.

<table>
<thead>
<tr>
<th>Policy granularity</th>
<th>From a space to an IP address range</th>
<th>From a source app to a destination app</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>For a space, org, or deployment</td>
<td>For app to app only</td>
</tr>
<tr>
<td>Traffic direction</td>
<td>Outbound control</td>
<td>Policies apply for incoming packets from other app instances</td>
</tr>
<tr>
<td><strong>Source app</strong></td>
<td>Is not known</td>
<td>Is identified because of direct addressability</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td><strong>Policies take affect</strong></td>
<td>After app restart</td>
<td>Immediately</td>
</tr>
</tbody>
</table>
Application Security Groups

This topic provides an overview of Application Security Groups (ASGs), and describes how to manage and administer them. Many of the steps below require the Cloud Foundry Command Line Interface (cf CLI) tool.

Note: If you are creating ASGs for the first time, see Restricting App Access to Internal PCF Components.

About Application Security Groups

Application Security Groups (ASGs) are a collection of egress rules that specify the protocols, ports, and IP address ranges where app or task instances send traffic.

ASGs define allow rules, and their order of evaluation is unimportant when multiple ASGs apply to the same space or deployment. The platform sets up rules to filter and log outbound network traffic from app and task instances. ASGs apply to both buildpack-based and Docker-based apps and tasks.

Staging and Running ASGs

Administrators can define a staging ASG for app and task staging, and a running ASG for app and task runtime.

When apps or tasks begin staging, they require traffic rules permissive enough to allow them to pull resources from the network. A running app or task no longer needs to pull resources, so traffic rules can be more restrictive and secure. To distinguish between these two security requirements, administrators can define a staging ASG for app and task staging with more permissive rules, and a running ASG for app and task runtime with less permissive rules.

Platform-Wide and Space-Scoped ASGs

To provide granular control when securing a deployment, administrators can assign platform-wide ASGs that apply to all app and task instances for the entire deployment, or space-scoped ASGs that apply only to apps and tasks in a particular space.

In environments with both platform-wide and space-specific ASGs, the ASGs for a particular space combine with the platform ASGs to determine the rules for that space.

Simplifying ASGs with a Services Subnet

ASGs can be complicated to configure correctly, especially when the specific IP addresses listed in a group change.

To simplify securing a deployment while still allowing apps reach external services, operators can deploy the services into a subnet that is separate from their Cloud Foundry deployment, then create ASGs for the apps that whitelist those service subnets, while denying access to any virtual machine (VM) hosting other apps.

For examples of typical ASGs, see the Typical Application Security Groups section of this topic.

Default ASGs

Pivotal Application Service (PAS) defines one default ASG, default_security_group. This group allows all outbound traffic from application containers on public and private networks except for the link-local range, 169.254.0.0/16, which is blocked.

warning: For security, PAS administrators must modify the default ASGs so that outbound network traffic cannot access internal components.

The ASG is defined in the Cloud Controller configuration as follows:

```
security_group_definitions:
- name: default_security_group
  rules:
  - protocol: all
```
ASG Sets

ASGs are applied by configuring ASG sets differentiated by scope, platform-wide or space specific, and lifecycle, staging or running.

Currently, four ASG sets exist in Cloud Foundry:

- Platform-wide staging ASG set, also called “default-staging”
- Platform-wide running ASG set, also called “default-running”
- Space-scoped staging ASG set
- Space-scoped running ASG set

In environments with both platform-wide and space-specific ASG sets, combine the ASG sets for a particular space with the platform ASG sets to determine the rules for that space.

The following table indicates the differences between the four sets.

<table>
<thead>
<tr>
<th>When an ASG is bound to the…</th>
<th>the ASG rules are applied to…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform-wide staging ASG set</td>
<td>the staging lifecycle for all apps and tasks.</td>
</tr>
<tr>
<td>Platform-wide running ASG set</td>
<td>the running lifecycle for all app and task instances.</td>
</tr>
<tr>
<td>Space-scoped staging ASG set</td>
<td>the staging lifecycle for apps and tasks in a particular space.</td>
</tr>
<tr>
<td>Space-scoped running ASG set</td>
<td>the running lifecycle for app and task instances in a particular space.</td>
</tr>
</tbody>
</table>

Typically, ASGs applied during the staging lifecycle are more permissive than the ASGs applied during the running lifecycle. This is because staging often requires access to different resources, such as dependencies.

You use different commands to apply an ASG to each of the four sets. For more information, see the Procedures section of this topic.

Note: To apply a staging ASG to apps within a space, you must use cf CLI v6.28.0 or later.

The Structure and Attributes of ASGs

ASG rules are specified as a JSON array of ASG objects. An ASG object has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>tcp, udp, icmp, or all</td>
<td>Required</td>
</tr>
<tr>
<td>destination</td>
<td>A single IP address, an IP address range like 192.0.2.0-192.0.2.50, or a CIDR block that can receive traffic.</td>
<td></td>
</tr>
<tr>
<td>ports</td>
<td>A single port, multiple comma-separated ports, or a single range of ports that can receive traffic. Examples: 443, 80,8080,8081, 8080-8081</td>
<td>Only possible if protocol is tcp or udp.</td>
</tr>
<tr>
<td>code</td>
<td>ICMP code</td>
<td>Required when protocol is icmp. A value of -1 allows all codes.</td>
</tr>
<tr>
<td>type</td>
<td>ICMP type</td>
<td>Required when protocol is icmp. A value of -1 allows all types.</td>
</tr>
<tr>
<td>log</td>
<td>Set to true to enable logging. For more information about how to configure system logs to be sent to a syslog drain, see the Using Log Management Services topic.</td>
<td>Logging is only supported with protocol type tcp</td>
</tr>
<tr>
<td>description</td>
<td>An optional text field for operators managing security group rules</td>
<td></td>
</tr>
</tbody>
</table>

Process for Administering ASGs

The following table outlines the flow of tasks that the administrator carries out over the lifecycle of ASGs. Procedures for each of these tasks are given in
Managing ASGs with the cf CLI below.

Note: If you are creating ASGs for the first time, see Restricting App Access to Internal PCF Components.

<table>
<thead>
<tr>
<th>Task</th>
<th>For more information, see</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Review the existing ASGs. If this is a new deployment, these consist of only the Default ASGs.</td>
<td>View ASGs</td>
</tr>
<tr>
<td>2. Create new ASGs.</td>
<td>Create ASGs</td>
</tr>
<tr>
<td>3. Update the existing ASGs.</td>
<td>Update ASGs</td>
</tr>
<tr>
<td>4. Bind ASGs to an ASG set.</td>
<td>Bind ASGs</td>
</tr>
<tr>
<td>5. If you need to delete an ASG, first unbind it, then delete it.</td>
<td>Unbind ASGs Delete ASGs</td>
</tr>
</tbody>
</table>

Managing ASGs with the cf CLI

This section provides the commands you need to create and manage ASGs.

View ASGs

Run the following cf CLI commands to view information about existing ASGs:

<table>
<thead>
<tr>
<th>Command</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf security-groups</td>
<td>All ASGs</td>
</tr>
<tr>
<td>cf staging-security-groups</td>
<td>All ASGs applied to the platform-wide staging ASG set</td>
</tr>
<tr>
<td>cf running-security-groups</td>
<td>All ASGs applied to the platform-wide running ASG set</td>
</tr>
<tr>
<td>cf security-group SECURITY-GROUP</td>
<td>All rules in the ASG named SECURITY-GROUP, for example, cf security-group dns</td>
</tr>
</tbody>
</table>

Note: You can also view ASGs in Apps Manager under the Settings tab of a space or an app.

Create ASGs

To create an ASG, perform the following steps:

1. Create a rules file: a JSON-formatted single array containing objects that describe the rules. See the following example, which allows ICMP traffic of code 0 and type 0 to all destinations, and TCP traffic to 10.0.11.0/24 on ports 80 and 443. Also see The Structure and Attributes of ASGs.

```json
[  
  {   
    "protocol": "icmp",   
    "destination": "0.0.0.0/0",   
    "type": 0,   
    "code": 0   
  },  
  {   
    "protocol": "tcp",   
    "destination": "10.0.11.0/24",   
    "ports": "80,443",   
    "log": true,   
    "description": "Allow http and https traffic from ZoneA"   
  }  
]
```

2. Run cf create-security-group SECURITY-GROUP PATH-TO-RULES-FILE. Replace SECURITY-GROUP with the name of your security group, and PATH-TO-RULES-FILE with the absolute or relative path to a rules file.

In the following example, my-asg is the name of a security group, and ~/workspace/my-asg.json is the path to a rules file.
After the ASG is created, you must bind it to an ASG set before it takes effect. See Bind ASGs below.

## Bind ASGs

**Note:** Binding an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

To apply an ASG, you must first bind it to an ASG set.

To bind an ASG to the platform-wide staging ASG set, run the `cf bind-staging-security-group SECURITY-GROUP` command. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
cf bind-staging-security-group my-asg
```

To bind an ASG to the platform-wide running ASG set, run the `cf bind-running-security-group SECURITY-GROUP` command. Replace `SECURITY-GROUP` with the name of your security group.

Example:

```
cf bind-running-security-group my-asg
```

To bind an ASG to a space-scoped running ASG set, run the `cf bind-security-group SECURITY-GROUP ORG SPACE` command. Replace `SECURITY-GROUP` with the name of your security group. Replace `ORG` and `SPACE` with the org and space where you want to bind the ASG set.

Example:

```
cf bind-security-group my-asg my-org my-space
```

To bind an ASG to a space-scoped staging ASG set, run the `cf bind-security-group SECURITY-GROUP ORG SPACE --lifecycle staging` command. Replace `SECURITY-GROUP` with the name of your security group. Replace `ORG` and `SPACE` with the org and space where you want to bind the ASG set.

Example:

```
cf bind-security-group my-asg my-org my-space
```

## Update ASGs

To update an existing ASG, perform the following steps.

1. Edit the ASG rules in the JSON file.

2. Run `cf update-security-group SECURITY-GROUP PATH-TO-RULES-FILE` command. Replace `SECURITY-GROUP` with the name of the existing ASG you want to change, and `PATH-TO-RULES-FILE` with the absolute or relative path to a rules file.

In the following example, `my-asg` is the name of a security group, and `~/workspace/my-asg-v2.json` is the path to a rules file.

```
cf update-security-group my-asg ~/workspace/my-asg-v2.json
```

**Note:** Updating an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

## Unbind ASGs

**Note:** Unbinding an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.
To unbind an ASG from the platform-wide staging ASG set, run

```
cf unbind-staging-security-group SECURITY-GROUP
```

Replace `SECURITY-GROUP` with the name of your security group.

Example:
```
$ cf unbind-staging-security-group my-asg
```

To unbind an ASG from the platform-wide running ASG set, run

```
cf unbind-running-security-group SECURITY-GROUP
```

Replace `SECURITY-GROUP` with the name of your security group.

Example:
```
$ cf unbind-running-security-group my-asg
```

To unbind an ASG from a specific space, run

```
cf unbind-security-group SECURITY-GROUP ORG SPACE --lifecycle running
```

Replace `SECURITY-GROUP` with the name of your security group. Replace `ORG` and `SPACE` with the org and space where you want to unbind the ASG set, and replace `running` with `staging` if you want to unbind from the staging ASG set.

Example:
```
$ cf unbind-security-group my-asg my-org my-space --lifecycle staging
```

Delete ASGs

**Note:** You can only delete unbound ASGs. To unbind ASGs, see Unbind ASGs above.

To delete an ASG, run

```
cf delete-security-group SECURITY-GROUP
```

Replace `SECURITY-GROUP` with the name of your security group.

Example:
```
$ cf delete-security-group my-asg
```

Typical ASGs

Below are examples of typical ASGs. Configure your ASGs in accordance with your organization’s network access policy for untrusted apps.

<table>
<thead>
<tr>
<th>ASG</th>
<th>For access to</th>
</tr>
</thead>
<tbody>
<tr>
<td>dns</td>
<td>DNS, either public or private</td>
</tr>
<tr>
<td>public-networks</td>
<td>Public networks, excluding IaaS metadata endpoints</td>
</tr>
<tr>
<td>private-networks</td>
<td>Private networks in accordance with RFC-1918</td>
</tr>
<tr>
<td>load-balancers</td>
<td>The internal Pivotal Application Service (PAS) load balancer and others</td>
</tr>
<tr>
<td>internal-proxies</td>
<td>Internal proxies</td>
</tr>
<tr>
<td>internal-databases</td>
<td>Internal databases</td>
</tr>
</tbody>
</table>

DNS

To resolve hostnames to IP addresses, apps require DNS server connectivity, which typically use port 53. Administrators should create or update a DNS ASG with appropriate rules. Administrators may further restrict the DNS servers to specific IP addresses or ranges of IP addresses.

Example
```
dns ASG:
```
Public Networks

Apps often require public network connectivity to retrieve app dependencies, or to integrate with services available on public networks. Example app dependencies include public Maven repositories, NPM, RubyGems, and Docker registries.

```json
[
  {   
    "protocol": "tcp",
    "destination": "0.0.0.0/0",
    "ports": [53]
  },
  {   
    "protocol": "udp",
    "destination": "0.0.0.0/0",
    "ports": [53]
  }
]
```

**Note:** You should exclude IaaS metadata endpoints, such as `169.254.169.254`, because the metadata endpoint can expose sensitive environment information to untrusted apps. The `public_networks` example below accounts for this recommendation.

Example `public_networks` ASG:

```json
[
  {   
    "destination": "0.0.0.0-9.255.255.255",
    "protocol": "all"
  },
  {   
    "destination": "11.0.0.0-169.253.255.255",
    "protocol": "all"
  },
  {   
    "destination": "169.255.0.0-172.15.255.255",
    "protocol": "all"
  },
  {   
    "destination": "172.32.0.0-192.167.255.255",
    "protocol": "all"
  },
  {   
    "destination": "192.169.0.0-255.255.255.255",
    "protocol": "all"
  }
]
```

Private Networks

Network connections that are commonly allowable in private networks include endpoints such as proxy servers, Docker registries, load balancers, databases, messaging servers, directory servers, and file servers. Configure appropriate private network ASGs as appropriate. You may find it helpful to use a naming convention with `private_networks` as part of the ASG name, such as `private_networks_databases`.

**Note:** You should exclude any private networks and IP addresses that app and task instances should not have access to.

Example `private_networks` ASG:
Marketplace Services

Each installed Marketplace Service requires its own set of ASG rules to function properly. See the installation instructions for each installed Marketplace Service to determine which ASG rules it requires. For more information about how to provision and integrate services, see the Services Overview topics.

About the ASG Creator Tool

The ASG Creator is a command line tool that you can use to create JSON rules files. The ASG Creator lets you specify IP addresses, CIDRs, and IP address ranges that you want to disallow traffic to, as well as the addresses that you want to allow traffic to. Based on these disallow/allow (exclude/include) lists that you provide as input, the ASG Creator formulates a JSON file of allow rules.

In turn, the JSON file is the input for the `cf create-security-group` command that creates an ASG.

You can download the latest release of the ASG Creator from the Cloud Foundry incubator repository on Github: https://github.com/cloudfoundry-incubator/asg-creator/releases/latest

ASG Logging

The KB article How to use Application Security Group (ASG) logging describes how you can use ASGs to correlate emitted logs back to an app.
GrootFS Disk Usage

This topic explains the concepts related to GrootFS disk space management in Pivotal Application Service (PAS).

GrootFS Stores

GrootFS is the container root filesystem management component for Garden. A container root filesystem orrootfs is often referred to as an image.

A GrootFS store is the directory in which rootsfs layers and container images are cached. This directory is configured by GrootFS and mounted on an XFS-formatted volume by the Garden job during BOSH VM creation.

Individual container root filesystems are provided via OverlayFS mounts.

Supplying GrootFS with an already formatted XFS volume for its store is not yet supported for BOSH-controlled deployments.

General Garbage Collection Behavior in GrootFS Stores

GrootFS stores are initialized to use the entirety of /var/vcap/data. If the reserved_space_for_other_jobs_in_mb is not set high enough, or if there are many images with few shared volumes, the store can use up all available space.

The thresholder component calculates and sets a value so that GrootFS’s garbage collector can attempt to ensure that a small reserved space is kept free for other jobs. GrootFS only tries to garbage collect when that threshold is reached. However, if all the rootsfs layers are actively in use by images, then garbage collection cannot occur and that space is used up.

Volumes

Underlying layers in rootsfs images are known as volumes in GrootFS. They are read-only and their changesets are layered together through an OverlayFS mount to create the root filesystems for containers.

When GrootFS writes each filesystem volume to disk, it also stores the number of bytes written to a file in a meta directory. The size of an individual volume is available in its corresponding metadata file. GrootFS also stores the SHA of each underlying volume used by an image in the meta folder.

For each container, GrootFS mounts the underlying volumes using overlay to a point in the images directory. This mount point is the rootsfs for the container and is read-write.

On disk, the read-write layer for each container can be found at /var/vcap/data/grootfs/store/unprivileged/images/CONTAINER-ID/diff (or /var/vcap/data/grootfs/store/privileged/images/CONTAINER-ID/diff for privileged containers.)

When GrootFS calls on the built-in XFS quota tooling to get disk usage for a container, it takes into account data written to those diff directories and not the data in the read-only volumes.

Volume Cleanup Example

When clean is called in GrootFS, any layers that are not being used by an existing rootsfs are deleted from the store. The cleanup only takes into account the volumes folders in the store.

For example, imagine that there are two rootsfs images from different base images, Image A and Image B:
They have a layer in common, layer-1. And after deleting Image B, layer-4 and layer-5 can be collected by clean, but not layer-1 because Image A still uses that layer.

For more information on how to calculate GrootFS disk usage in your deployment, see Examining GrootFS Disk Usage.

Additional Information

For more information, see the following sections of [garden-runc-release](#):

- overlay-xfs-setup
- grootfs-utils
- thresholder
Cloud Foundry Components

Page last updated:

Cloud Foundry components include a self-service application execution engine, an automation engine for application deployment and lifecycle management, and a scriptable command line interface (CLI), as well as integration with development tools to ease deployment processes. Cloud Foundry has an open architecture that includes a buildpack mechanism for adding frameworks, an application services interface, and a cloud provider interface.

See the descriptions below for more information about Cloud Foundry components. Some descriptions include links to more detailed documentation.

Routing

Router

The router routes incoming traffic to the appropriate component, either a Cloud Controller component or a hosted application running on a Diego Cell.

The router periodically queries the Diego Bulletin Board System (BBS) to determine which cells and containers each application currently runs on. Using this information, the router recomputes new routing tables based on the IP addresses of each cell virtual machine (VM) and the host-side port numbers for the cell’s containers.

Authentication

OAuth2 Server (UAA) and Login Server

The OAuth2 server (the UAA) and Login Server work together to provide identity management.

App Lifecycle

Cloud Controller and Diego Brain

The Cloud Controller (CC) directs the deployment of applications. To push an app to Cloud Foundry, you target the Cloud Controller. The Cloud Controller then directs the Diego Brain through the CC-Bridge components to coordinate individual Diego cells to stage and run applications.

The Cloud Controller also maintain records of orgs, spaces, user roles, services, and more.
nsync, BBS, and Cell Reps

To keep applications available, cloud deployments must constantly monitor their states and reconcile them with their expected states, starting and stopping processes as required.

The nsync, BBS, and Cell Rep components work together along a chain to keep apps running. At one end is the user. At the other end are the instances of applications running on widely-distributed VMs, which may crash or become unavailable.

Here is how the components work together:

- **nsync** receives a message from the Cloud Controller when the user scales an app. It writes the number of instances into a *DesiredLRP* structure in the Diego BBS database.
- **BBS** uses its convergence process to monitor the *DesiredLRP* and *ActualLRP* values. It launches or kills application instances as appropriate to ensure the *ActualLRP* count matches the *DesiredLRP* count.
- **Cell Rep** monitors the containers and provides the *ActualLRP* value.

App Storage and Execution

Blobstore

The blobstore is a repository for large binary files, which Github cannot easily manage because GitHub is designed for code. The blobstore contains the following:

- Application code packages
- Buildpacks
- Droplets

You can configure the blobstore as either an internal server or an external S3 or S3-compatible endpoint. See this Knowledge Base article for more information about the blobstore.

Diego Cell

Application instances, application tasks, and staging tasks all run as Garden containers on the Diego Cell VMs. The Diego cell rep component manages the lifecycle of those containers and the processes running in them, reports their status to the Diego BBS, and emits their logs and metrics to Loggregator.

Services

Service Brokers

Applications typically depend on services such as databases or third-party SaaS providers. When a developer provisions and binds a service to an application, the service broker for that service is responsible for providing the service instance.
Messaging

Internal HTTPS and BBS

Cloud Foundry component VMs communicate with each other internally through HTTP and HTTPS protocols, sharing temporary messages and data stored in Diego’s Bulletin Board System (BBS).

- BOSH Director collocates a BOSH DNS server on every deployed VM. All VMs keep up-to-date DNS records for all the other VMs in the same foundation, enabling service discovery between VMs. BOSH DNS also provides client-side load-balancing by randomly selecting a healthy VM when multiple VMs are available.
- Diego’s Bulletin Board System (BBS) stores more frequently updated and disposable data such as cell and app status, unallocated work, and heartbeat messages, as well as longer-lived distributed locks. The BBS stores data in MySQL, using the Go MySQL Driver.

The route-emitter component uses the NATS protocol to broadcast the latest routing tables to the routers.

Metrics and Logging

Loggregator

The Loggregator system streams application logs to developers.
**Cloud Controller**

Page last updated:

The Cloud Controller provides REST API endpoints for clients to access the system. The Cloud Controller maintains a database with tables for orgs, spaces, services, user roles, and more.

**Diego Auction**

The Cloud Controller uses the Diego Auction to balance application processes over the cells in a Cloud Foundry installation.

**Database (CC_DB)**

The Cloud Controller database has been tested with MySQL.

**Blobstore**

To stage and run apps, Cloud Foundry manages and stores the following types of binary large object (blob) files:

<table>
<thead>
<tr>
<th>Blob Type</th>
<th>Description</th>
<th>Location in Blobstore</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Packages</td>
<td>Full contents of app directories, including source code and resource files,</td>
<td>/cc-packages</td>
</tr>
<tr>
<td></td>
<td>zipped into single blob files.</td>
<td></td>
</tr>
<tr>
<td>Buildpacks</td>
<td>Buildpack directories, which Diego cells download to compile and stage apps</td>
<td>/cc-buildpacks</td>
</tr>
<tr>
<td></td>
<td>with.</td>
<td></td>
</tr>
<tr>
<td>Resource Cache</td>
<td>Large files from app packages that the Cloud Controller stores with a SHA</td>
<td>/cc-resources</td>
</tr>
<tr>
<td></td>
<td>for later re-use. To save bandwidth, the Cloud Foundry Command Line Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(cf CLI) only uploads large application files that the Cloud Controller has</td>
<td></td>
</tr>
<tr>
<td></td>
<td>not already stored in the resource cache.</td>
<td></td>
</tr>
<tr>
<td>Buildpack Cache</td>
<td>Large files that buildpacks generate during staging, stored for later re-use</td>
<td>/cc-droplets/buildpack_cache</td>
</tr>
<tr>
<td></td>
<td>. This cache lets buildpacks run more quickly when staging apps that have</td>
<td></td>
</tr>
<tr>
<td></td>
<td>been staged previously.</td>
<td></td>
</tr>
<tr>
<td>Droplets</td>
<td>Staged apps, packaged with everything needed to run in a container.</td>
<td>/cc-droplets</td>
</tr>
</tbody>
</table>

Cloud Foundry blobstores use the Fog Ruby gem to store blobs in services like Amazon S3, WebDAV, or the NFS filesystem. The file system location of an internal blobstore is `/var/vcap/store/shared`.

A single blobstore typically stores all five types of blobs, but you can configure the Cloud Controller to use separate blobstores for each type.

**Automatic Blob Cleanup**

After a blob deletion fails silently or something else goes wrong, the blobstore may contain blobs that the Cloud Controller no longer needs or lists in its database. These are called orphan blobs, and they waste blobstore capacity.

The Cloud Controller detects and removes orphan blobs by scanning part of the blobstore daily and checking for any blobs that its database does not account for. The process scans through the entire blobstore every week, and only removes blobs that show as orphans for three consecutive days.

The Cloud Controller performs this automatic cleanup when the `cloud_controller_worker` job property `cc.perform_blob_cleanup` is set to `true`.

**Manual Blob Cleanup**

The Cloud Controller does not track resource cache and buildpack cache blob types in its database, so it does not clean them up automatically as it does with app package, buildpack, and droplet type blobs.
To clean up the buildpack cache, admin users can run:

```
cf curl -X DELETE
/v2/blobstores/buildpack_cache
```
This empties the buildpack cache completely, which is a safe operation.

To clean up the resource cache, delete it as follows:

- **Internal blobstore**: Run `bosh ssh` to connect to the blobstore VM (NFS or WebDav) and `rm *` the contents of the `/var/vcap/store/shared/cc-resources` directory.
- **External blobstore**: Use the file store's API to delete the contents of the `resources` bucket.

Do not manually delete app package, buildpack, or droplet blobs from the blobstore. To free up resources from those locations, run `cf delete-buildpack` for buildpacks or `cf delete` for app packages and droplets.

### Testing

By default, `rspec` runs a test suite with the SQLite in-memory database. Specify a connection string using the `DB_CONNECTION` environment variable to test against MySQL. For example:

```
DB_CONNECTION="mysql2://root:password@localhost:3306/ccng" rspec
```
This information was adapted from the NATS README. NATS is a lightweight publish-subscribe and distributed queueing messaging system written in Ruby. Ops Manager sends all NATS traffic using Transport Layer Security (TLS) encryption by default.

Install NATS

```
$ gem install nats
# or
$ rake geminstall
$ nats-sub foo &
$ nats-pub foo 'Hello World!'  
```

Basic Usage

```ruby
require "nats/client"

NATS.start do
  # Simple Subscriber
  NATS.subscribe('foo') { |msg| puts "Msg received: '#{msg}'" }

  # Simple Publisher
  NATS.publish('foo.bar.baz', 'Hello World!')

  # Unsubscribing
  sid = NATS.subscribe('bar') { |msg| puts "Msg received: '#{msg}'" }
  NATS.unsubscribe(sid)

  # Requests
  NATS.request('help') { |response| puts "Got a response: '#{response}'" }

  # Replies
  NATS.subscribe('help') { |msg, reply| NATS.publish(reply, "I'll help!") }

  # Stop using NATS.stop, exits EM loop if NATS.start started the loop
  NATS.stop
end
```

Wildcard Subscriptions

```ruby
# "***" matches any token, at any level of the subject.
NATS.subscribe('foo.*') { |msg, reply, sub| puts "Msg received on [#{sub}]: '#{msg}''
NATS.subscribe('foo.bar.*') { |msg, reply, sub| puts "Msg received on [#{sub}]: '#{msg}'" }
NATS.subscribe('*.bar.*') { |msg, reply, sub| puts "Msg received on [#{sub}]: '#{msg}'" }

# ">" matches any length of the tail of a subject and can only be the last token
# E.g. 'foo.>' will match 'foo.bar', 'foo.bar.baz', 'foo.foo.bar.baz.22'
NATS.subscribe('foo.>') { |msg, reply, sub| puts "Msg received on [#{sub}]: '#{msg}'" }
```

Queues Groups

```ruby
# All subscriptions with the same queue name will form a queue group
# Each message will be delivered to only one subscriber per queue group, queueing semantics
# You can have as many queue groups as you want
# Normal subscribers will continue to work as expected.
NATS.subscribe(subject, queue => 'job.workers') { |msg| puts "Received '#{msg}'" }
```

© Copyright Pivotal Software Inc, 2013-2019
Advanced Usage

```ruby
# Publish with closure, callback fires when server has processed the message
NATS.publish('foo', 'You done?') { puts 'msg processed!' }

# Timeouts for subscriptions
sid = NATS.subscribe('foo') { received += 1 }
NATS.timeout(sid, TIMEOUT_IN_SECS) { timeout_recv = true }

# Timeout unless a certain number of messages have been received
NATS.timeout(sid, TIMEOUT_IN_SECS, expected => 2) { timeout_recv = true }

# Auto-unsubscribe after MAX_WANTED messages received
NATS.unsubscribe(sid, MAX_WANTED)

# Multiple connections
NATS.subscribe('test') do |msg|
  puts 'received msg'
  NATS.stop
end

# Form second connection to send message on
NATS.connect { NATS.publish('test', 'Hello World!') }
```
Gorouter routes traffic coming into Cloud Foundry to the appropriate component, whether the request comes from an operator addressing the Cloud Controller or from an application user accessing an app running on a Diego Cell. Handling both platform and app requests with the same process centralizes routing logic and simplifies support for WebSockets and other types of traffic (for example, through HTTP CONNECT).

Refer to the following instructions for help getting started with Gorouter in a standalone environment.

**Setup**

```
$ git clone https://github.com/cloudfoundry/gorouter.git
$ cd gorouter
$ git submodule update --init
$ ./bin/go install gorouter/gorouter
$ gem install nats
```

**Start**

```
# Start NATS server in daemon mode
$ nats-server -d

# Start gorouter
$ ./bin/gorouter
```

**Usage**

Gorouter receives route updates through NATS. By default, routes that have not been updated in two minutes are pruned. Therefore, to maintain an active route, you must ensure that the route is updated at least every two minutes. The format of these route updates is as follows:

```
{
  "host": "127.0.0.1",
  "port": 4567,
  "uris": [
    "my_first_url.vcap.me",
    "my_second_url.vcap.me"
  ],
  "tags": {
    "another_key": "another_value",
    "some_key": "some_value"
  }
}
```

Such a message can be sent to both the `gorouter.register` subject to register URIs, and to the `gorouter.unregister` subject to unregister URIs, respectively.

```
$ nhttp ruby -e \"sinatra \{- get \"/\" \{ "Hello!" \} \} & \n\$ nats-pub 'gorouter.register' \"{"host":"127.0.0.1","port":4567, 
"uris":["my_first_url.vcap.me","my_second_url.vcap.me"],
"tags": {"another_key":"another_value","some_key":"some_value"}\} \nPublished [gorouter.register] : "{'host":"127.0.0.1","port":4567, 
"uris":["my_first_url.vcap.me","my_second_url.vcap.me"],
"tags": {"another_key":"another_value","some_key":"some_value"}\} 
```

**Instrumentation**

Gorouter provides `/varz` and `/healthz` http endpoints for monitoring.
The `/routes` endpoint returns the entire routing table as JSON. Each route has an associated array of `host:port` entries.

All of the endpoints require HTTP basic authentication, credentials for which you can acquire through NATS. You can explicitly set the `port`, `user`, and `pass` (is the config attribute) in the `gorouter.yml` config file `status` section.

Example interaction with `curl`:

```
$ curl -vvv "http://someuser:somepass@127.0.0.1:8080/routes"
* About to connect() to 127.0.0.1 port 8080 (0)
* Trying 127.0.0.1... (0)
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (9)
* Server auth using Basic with user 'someuser'
> GET /routes HTTP/1.1
> Authorization: Basic c29tZXVzZXI6c29tZXBhc3M=
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL/0.9.8r zlib/1.2.5
> Host: 127.0.0.1:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Mon, 25 Mar 2013 20:31:27 GMT
< Transfer-Encoding: chunked
< "["0295dd314aaf582f201e655cbd74ade5.cloudfoundry.me": ["127.0.0.1:34567"],
  "03e316d6aa375d1dc1153700da5f1798.cloudfoundry.me": ["127.0.0.1:34568"]}
```

**Logs**

This section provides details about Gorouter logging.

**Levels**

The following table describes the log levels supported by Gorouter. The log level is set to `debug` and is not configurable.

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>fatal</td>
<td>Gorouter is unable to handle any requests due to a fatal error.</td>
<td>Gorouter cannot bind to its TCP port, a CF component has published invalid data to Gorouter.</td>
</tr>
<tr>
<td>error</td>
<td>An unexpected error has occurred.</td>
<td>Gorouter failed to fetch token from UAA service.</td>
</tr>
<tr>
<td>info</td>
<td>An expected event has occurred.</td>
<td>Gorouter started or exited, Gorouter has begun to prune routes for stale droplets.</td>
</tr>
<tr>
<td>debug</td>
<td>A lower-level event has occurred.</td>
<td>Route registration, route unregistration.</td>
</tr>
</tbody>
</table>

**Message Contents**

This section provides a sample Gorouter log message and an explanation of the contents.

```
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_level</td>
<td>Logging level of the message</td>
</tr>
<tr>
<td>timestamp</td>
<td>Epoch time of the log</td>
</tr>
<tr>
<td>message</td>
<td>Content of the log line</td>
</tr>
<tr>
<td>source</td>
<td>Gorouter function that initiated the log message</td>
</tr>
<tr>
<td>data</td>
<td>Additional information that varies based on the message</td>
</tr>
</tbody>
</table>
About Access Logs

Access logs provide information for the following fields when receiving a request:

- `<Request Host>` - `[<Start Date>]` - `<Request Method>` - `<Request URL>` - `<Request Protocol>` - `<Status Code>` - `<Bytes Received>` - `<Bytes Sent>` - `<Referer>` - `<User-Agent>` - `<Remote Address>`

The following are optional fields: `<Status Code>`, `<Response Time>`, `<Application ID>`, `<Application Index>`, and `<Extra Headers>`. If the access log lacks a `<Status Code>`, `<Response Time>`, `<Application ID>`, or `<Application Index>`, the corresponding field shows `-`.

Note: Access logs are also redirected to syslog.
User Account and Authentication (UAA) Server

This topic provides an overview of the User Account and Authentication (UAA) Server, the identity management service for Cloud Foundry.

The primary role of the UAA is as an OAuth2 provider, issuing tokens for client apps to use when they act on behalf of Cloud Foundry users. In collaboration with the login server, the UAA can authenticate users with their Cloud Foundry credentials, and can act as an SSO service using those, or other, credentials.

The UAA has endpoints for managing user accounts and for registering OAuth2 clients, as well as various other management functions.

Quick Start

You can deploy the UAA locally or to Cloud Foundry.

- Deploy UAA Locally
- Deploy UAA to Cloud Foundry

Deploy UAA Locally

Follow the instructions below to deploy and run the UAA locally.

1. In a terminal window, clone the UAA GitHub repository.

   ```bash
   $ git clone git://github.com/cloudfoundry/uaa.git
   ```

2. Navigate to the directory where you cloned the UAA GitHub repository, then run the `./gradlew run` command to build and run all the components that comprise the UAA and the example programs, `uaa`, `samples/api`, and `samples/app`.

   ```bash
   $ cd uaa
   $ ./gradlew run
   ```

3. If successful, the three apps run together on a single instance of Tomcat listening on port 8080, with endpoints `/uaa`, `/app`, and `/api`.

Use Local UAA

Follow the steps below to access and use a locally-deployed UAA server.

1. Run the UAA server as described in the Deploy Locally section, above.

2. Open another terminal window. From the project base directory, run `curl localhost:8080/uaa/info -H "Accept: application/json"` to confirm the UAA is running. You should see basic information about the system. For example:
3. Run 

```bash
$ gem install cf-uaac
```
to install the Cloud Foundry UAA Command Line Client (UAAC) Ruby gem.

4. Run 

```bash
$ uaac target http://localhost:8080/uaa
```
to target the local UAA server endpoint.

5. Run 

```bash
$ uaac token client get CLIENT_NAME -s CLIENT_SECRET
```
to obtain an access token. Replace `CLIENT_NAME` and `CLIENT_SECRET` with actual values. For example, when starting up the UAA locally for development there should be a predefined `admin client` you can use:

```bash
$ uaac token client get admin -s adminsecret
```
If you do not specify the client secret, UAAC will show an interactive prompt where you must enter the client secret value.

The `uaac token client get` command requests an access token from the server using the OAuth2 `client credentials` grant type.

6. View your UAAC token context. When UAAC obtains a token, the token and other metadata is stored in the `.uaac.yml` file on your local machine. To view the token you have obtained, use the command `uaac context`. For example:

```bash
$ uaac context
```

Copy the access token from this output for the following step.

7. Run 

```bash
$ uaac token decode ACCESS-TOKEN-VALUE
```
to view information in the token, which is encoded using the JSON Web Token (JWT) format. Replace `ACCESS-TOKEN-VALUE` with your access token, copied from the `uaac context` output. The UAAC should display all the claims inside the token body. For example:
Deploy UAA to Cloud Foundry

Follow the instructions below to build the UAA as an app and push it to Cloud Foundry using the Cloud Foundry Command Line Interface (cf CLI).

1. In a terminal window, clone the UAA GitHub repository.

   ```bash
   $ git clone git://github.com/cloudfoundry/uaa.git
   ```

2. Navigate to the directory where you cloned the UAA GitHub repository, then run the 
   `./gradlew :cloudfoundry-identity-uaa:war` command build the UAA as a WAR file.

   ```bash
   $ cd uaa
   $ ./gradlew :cloudfoundry-identity-uaa:war
   ```

3. Run the `cf` CLI `cf push APP-NAME -m 512M -p PATH-TO-WAR-FILE --no-start` command to push the app to Cloud Foundry. Replace `APP-NAME` with a name for your UAA app, and `PATH-TO-WAR-FILE` with the path to the WAR file you created in the previous step. For example:

   ```bash
   $ cf push MYUAA -m 512M -p uaa/build/libs/cloudfoundry-identity-uaa-1.8.0.war --no-start
   ```

4. Run `cf set-env APP-NAME SPRING_PROFILES_ACTIVE default` to set the `SPRING_PROFILES_ACTIVE` environment variable with the value `default`. Replace `APP-NAME` with the name of your app that you used in the previous step. For example:

   ```bash
   $ cf set-env MYUAA SPRING_PROFILES_ACTIVE default
   ```

5. Run `cf start APP-NAME` to start your app. Replace `APP-NAME` with the name of your app. For example:

   ```bash
   $ cf start MYUAA
   ```

Use Remote UAA

You use a UAA server that you pushed as an app to Cloud Foundry in a similar way to one you run locally. You do not need app token encoding because you do not have the client secret.

Follow the steps below to access and use a UAA server that you pushed as an app to Cloud Foundry.

1. **Deploy UAA to Cloud Foundry** as described above.

2. From the project base directory, run `curl -H "Accept: application/json" APP-FQDN/login` to query the external login endpoint about the system. Replace `APP-FQDN` with the FQDN of your app. For example:
1. Request a token from the UAA server using:

   ```
   curl -H "Accept: application/json" uaa.example.org/login
   ```

2. The response contains the necessary information:

   ```
   "timestamp": "2014-09-15T18:25:04+0000",
   "app": {"version": "1.8.3"},
   "commit_id": "git-metadata-not-found",
   "prompts": {"username": ["text", "Email"],
   "password": ["password", "Password"]
   }
   ```

3. Run `gem install cf-uaac` to install the Cloud Foundry UAA Command Line Client (UAAC) Ruby gem.

   ```
   $ gem install cf-uaac
   ```

4. Run `uaac target APP-FQDN` to target the remote UAA Server endpoint. Replace `APP-FQDN` with the FQDN of your app.

   ```
   $ uaac target uaa.example.org
   ```

5. Run `uaac token get USERNAME PASSWORD` to log in. Replace `USERNAME` with your user name, and `PASSWORD` with your password. For example:

   ```
   $ uaac token get marissa koala
   ```

   If you do not specify a username and password, the UAAC prompts you to supply them.

Integration Tests

Run the integration tests with the following command:

```
$ ./gradlew integrationTest
```

This command starts a UAA server running in a local Apache Tomcat instance. By default, the service URL is set to `http://localhost:8080/uaa`

You can set the environment variable `CLOUD_FOUNDRY_CONFIG_PATH` to a directory containing a `uaa.yml` file where you change the URLs used in the tests, and where you can set the UAA server context root.

Custom YAML Configuration

Follow the steps below to modify the runtime parameters.

1. Create a `uaa.yml` file in the following format:

   ```yaml
   uaa:
   host: UAA-HOSTNAME
   test:
   username: USERNAME
   password: PASSWORD
   email: EMAIL-ADDRESS
   ```

   Replace the values in the above format as follows:

   - **UAA-HOSTNAME**: The FQDN of UAA app. Example: `uaa.example.org`
   - **USERNAME**: A valid username. Example: `dev@example.org`
   - **PASSWORD**: Password for the above username.
   - **EMAIL-ADDRESS**: Email address for the above user. Example: `dev@example.org`

2. From the `uaa/uaa` directory, run `CLOUD_FOUNDRY_CONFIG_PATH=/tmp ./gradlew test`.

3. The web app looks for a YAML file in the following locations when it starts, with later entries overriding earlier ones:
Test with PostgreSQL or MySQL

The default UAA unit tests, `./gradlew test`, use HyperSQL (hsqldb).

To use a different database management system, create a `uaa.yml` file containing `spring_profiles: default,OTHER-DBMS` in the `src/main/resources` directory. Replace `OTHER-DBMS` with the name of the other database management system to use.

For example, run the following command to run the unit tests using PostgreSQL instead of hsqldb:

```
$ echo "spring_profiles: default,postgresql" > src/main/resources/uaa.yml
$ ./gradlew test integrationTest
```

Run the following command to run the unit tests using MySQL instead of hsqldb:

```
$ echo "spring_profiles: default,mysql" > src/main/resources/uaa.yml
$ ./gradlew test integrationTest
```

You can find the database configuration for the common and scim modules at `common/src/test/resources/(mysql|postgresql).properties` and `scim/src/test/resources/(mysql|postgresql).properties`.

UAA Projects

The following UAA projects exist:

- **common**: A module containing a JAR with all the business logic. `common` is used in the web apps listed below.
- **uaa**: The UAA server. `uaa` provides an authentication service and authorized delegation for back-end services and apps by issuing OAuth2 access tokens.
- **api**: A sample OAuth2 resource service that returns a mock list of deployed apps. `api` provides resources that other apps might want to access on behalf of the resource owner.
- **app**: A sample user app that uses both `api` and `uaa`. `app` is a web app that requires single sign-on and access to the `api` service on behalf of users.
- **scim**: The SCIM user management module used by UAA.

UAA Server

The authentication service, `uaa`, is a Spring MVC web app. You can deploy it in Tomcat or your container of choice, or execute `./gradlew run` to run it directly from the `uaa` directory in the source tree. When run with Gradle, `uaa` listens on port 8080 and has URL `http://localhost:8080/uaa`.

The UAA Server supports the APIs defined in the UAA-APIs document which include the following:

- **The OAuth2** `/authorize` and `/token` endpoints
- A `/login_info` endpoint to allow querying for required login prompts
- A `/check_token` endpoint to allow resource servers to obtain information about an access token submitted by an OAuth2 client
- SCIM user provisioning endpoint
- OpenID connect endpoints to support authentication `/userinfo` and `/check_id`

Command line clients can perform authentication by submitting credentials directly to the `/authorize` endpoint.

An `ImplicitAccessTokenProvider` exists in Spring Security OAuth to use if your client is Java.

By default, `uaa` will launch with a context root `/uaa`. 

© Copyright Pivotal Software Inc, 2013-2019
Configuration

A `uaa.yml` file exists in the app. This file provides defaults to the placeholders in the Spring XML.

You can override any occurrences of `${placeholder.name}` in the XML by adding it to the `uaa.yml` file, or by providing a System property, `-D` to JVM, with the same name.

All passwords and client secrets in the config files are in plain text, but are inserted into the UAA database encrypted with BCrypt.

User Account Data

The default uses an in-memory RDBMS user store, pre-populated with a single test user `marissa` with password `koala`.

To use PostgreSQL for user data, activate the Spring profile `postgresql`.

You can configure the active profiles in `uaa.yml` using the following:

```
spring_profiles: postgresql,default
```

To use PostgreSQL instead of HyperSQL (hsqldb):

```
$ echo "spring_profiles: postgresql,default" > src/main/resources/uaa.yml
$ ./gradlew run
```

To bootstrap a microcloud-type environment, you need an admin client. A database initializer component that inserts an admin client exists. If the default profile is active, for example not `postgresql`, a cf CLI client exists so that the Gem login works with no additional configuration required. You can override the default settings and add additional clients in the `uaa.yml` file:

```
oauth:
  clients:
    admin:
      authorized-grant-types: client_credentials
      scope: read,write,password
      authorities: ROLE_CLIENT,ROLE_ADMIN
      id: admin
      secret: adminclientsecret
      resource-ids: clients
```

You can use the admin client to create additional clients. You must have a client with read/write access to the `scim` resource to create user accounts. The integration tests handle this automatically by inserting client and user accounts as necessary for the tests.

Sample Apps

Two sample apps are included with the UAA: `api` and `app`.

Run them with `./gradlew run` from the `uaa` root directory. All three apps, `/uaa`, `/api`, and `/app`, are simultaneously deployed.

API Sample App

The `api` sample app is an example resource server. It hosts a service that returns a list of mock apps under `/apps`.

APP Sample App

The `app` sample app is a user interface app, primarily aimed at browsers, that uses OpenID Connect for authentication and OAuth2 for access grants. `app` authenticates with the Auth service, then accesses resources in the API service. Run it with `./gradlew run` from the `uaa` root directory.

The app can operate in multiple different profiles according to the location and presence of the UAA server and the login app. By default, the app looks for a UAA on `localhost:8080/uaa`, but you can change this by setting the `UAA_PROFILE` environment variable or System Property.

© Copyright Pivotal Software Inc, 2013-2019 765 2.5
The app source code, `samples/app/src/main/resources`, contains multiple properties files pre-configured with different likely locations for those servers. The names of these properties files follow the form `app-UAA_PROFILE.properties`.

The naming convention for the `UAA_PROFILE` is as follows:

- **local**: a localhost deployment
- **vcap**: a `vcap.me` deployment
- **staging**: a staging deployment

Profile names can be hyphenated to indicate multiple contexts. For example, `local-vcap` can be used when the login server is in a different location than the UAA server.

**Use Cases**

1. See all apps:

   ```
   GET /app/apps
   ```

   Browser is redirected through a series of authentication and access grant steps, and then the photos are shown.

2. See the currently logged in user details, a selection of attributes from the OpenID provider:

   ```
   GET /app
   ```

**LOGIN App**

The `login` app is a user interface for authentication. The UAA can also authenticate user accounts, but only if it manages them itself, and it only provides a basic UI. You can brand and customize the login app for non-native authentication and for more complicated UI flows, like user registration and password reset.

The login app is itself an OAuth2 endpoint provider, but delegates those features to the UAA server. Therefore, configuration for the login app consists of locating the UAA through its OAuth2 endpoint URLs and registering the login app itself as a client of the UAA. A `login.yml` for the UAA locations exists, such as for a local `vcap` instance:

```yaml
uaa:
  url: http://uaa.vcap.example.net
  token:
    url: http://uaa.vcap.example.net/oauth/token
  login:
    url: http://uaa.vcap.example.net/login.do
```

An environment variable or Java System property also exists, `LOGIN_SECRET`, for the client secret that the app uses when it authenticates itself with the UAA. The login app is registered by default in the UAA only if there are no active Spring profiles. In the UAA, the registration is located in the `oauth-clients.xml` config file:

```xml
<oauth2-client id="login" secret="loginscret">
  <authorized-grant-types>client_credentials</authorized-grant-types>
  <authorities>ROLE_LOGIN</authorities>
  <resource-ids>oauth</resource-ids>
</oauth2-client>
```

**Use Cases**

1. Authenticate:

   ```
   GET /login
   ```

   The sample app presents a form login interface for the backend UAA, and an OpenID widget where a user can authenticate using Google or other credentials.

2. Approve OAuth2 token grant:
GET /oauth/authorize?client_id=app&response_type=code...

Standard OAuth2 Authorization Endpoint. The UAA handles client credentials and all other features in the back end, and the login app is used to render the UI.

3. Obtain access token:

POST /oauth/token

Standard OAuth2 Authorization Endpoint passed through to the UAA.

Scopes

UAA covers multiple scopes of privilege, including access to UAA, access to Cloud Controller, and access to the router.

See the tables below for a description of the scopes covered by UAA:

- **UAA Scopes**
- **Cloud Controller Scopes**
- **Router Scopes**
- **Other Scopes**

### UAA Scopes

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uaa.user</td>
<td>This scope indicates that this is a user. It is required in the token if submitting a GET request to the OAuth 2 /authorize endpoint.</td>
</tr>
<tr>
<td>uaa.none</td>
<td>This scope indicates that this client will not be performing actions on behalf of a user.</td>
</tr>
<tr>
<td>uaa.admin</td>
<td>This scope indicates that this is the superuser.</td>
</tr>
<tr>
<td>scim.write</td>
<td>This scope gives admin write access to all SCIM endpoints, /Users, and /Groups.</td>
</tr>
<tr>
<td>scim.read</td>
<td>This scope gives admin read access to all SCIM endpoints, /Users, and /Groups.</td>
</tr>
<tr>
<td>scim.create</td>
<td>This scope gives the ability to create a user with a POST request to the /Users endpoint, but not to modify, read, or delete users.</td>
</tr>
<tr>
<td>scim.userids</td>
<td>This scope is required to convert a username and origin into a user ID and vice versa.</td>
</tr>
<tr>
<td>scim.invite</td>
<td>This scope is required to participate in invitations using the /invite_users endpoint.</td>
</tr>
<tr>
<td>groups.update</td>
<td>This scope gives the ability to update a group. This ability can also be provided by the broader scim.write scope.</td>
</tr>
<tr>
<td>password.write</td>
<td>This admin scope gives the ability to change a user's password.</td>
</tr>
<tr>
<td>openId</td>
<td>This scope is required to access the /userinfo endpoint. It is intended for OpenID clients.</td>
</tr>
<tr>
<td>idps.read</td>
<td>This scope gives read access to retrieve identity providers from the /identity-providers endpoint.</td>
</tr>
<tr>
<td>idps.write</td>
<td>This scope gives the ability to create and update identity providers from the /identity-providers endpoint.</td>
</tr>
<tr>
<td>clients.admin</td>
<td>This scope gives the ability to create, modify, and delete clients.</td>
</tr>
<tr>
<td>clients.write</td>
<td>This scope gives the ability to create a client that has scopes with the testclient namespace. Authorities are limited to uaa.resource.</td>
</tr>
<tr>
<td>clients.read</td>
<td>This scope gives the ability to read information about clients.</td>
</tr>
<tr>
<td>clients.secret</td>
<td>This admin scope is required to change the password of a client.</td>
</tr>
<tr>
<td>zones.read</td>
<td>This scope is required to invoke the /identity-zones endpoint to read identity zones.</td>
</tr>
<tr>
<td>zones.write</td>
<td>This scope is required to invoke the /identity-zones endpoint to create and update identity zones.</td>
</tr>
<tr>
<td>scim.zones</td>
<td>This is a limited scope that only allows adding a user to, or removing a user from, zone management groups under the path /Groups/zones.</td>
</tr>
</tbody>
</table>
OAuth Scopes

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/approvals endpoint</td>
<td>This scope is required to approve or reject clients to act on a user’s behalf. This is a default scope defined in the <code>uaa.yml</code> file.</td>
</tr>
<tr>
<td>/approvals</td>
<td>This scope is not currently used.</td>
</tr>
<tr>
<td>/approvals</td>
<td>This scope indicates that this is a resource server, used for the <code>/check_token</code> endpoint.</td>
</tr>
<tr>
<td>uaa.resource</td>
<td>This scope indicates that this is a resource server.</td>
</tr>
<tr>
<td>zones.ZONE-ID.admin</td>
<td>This scope permits operations in a designated zone, such as creating identity providers or clients in another zone, by authenticating against the default zone. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.read</td>
<td>This scope translates into <code>clients.admin</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.clients.admin</td>
<td>This scope translates into <code>clients.admin</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.clients.read</td>
<td>This scope translates into <code>clients.read</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.clients.write</td>
<td>This scope translates into <code>clients.write</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.clients.scim.read</td>
<td>This scope translates into <code>scim.read</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.clients.scim.create</td>
<td>This scope translates into <code>scim.create</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.clients.scim.write</td>
<td>This scope translates into <code>scim.write</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
<tr>
<td>zones.ZONE-ID.idps.read</td>
<td>This scope translates into <code>idps.read</code> after zone switch completes. This scope is used with the X-Identity-Zone-Id header.</td>
</tr>
</tbody>
</table>

Cloud Controller Scopes

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller.read</td>
<td>This scope gives the ability to read from any Cloud Controller route the token has access to.</td>
</tr>
<tr>
<td>cloud_controller.write</td>
<td>This scope gives the ability to post to Cloud Controller routes the token has access to.</td>
</tr>
<tr>
<td>cloud_controller.admin</td>
<td>This admin scope gives full permissions to Cloud Controller.</td>
</tr>
<tr>
<td>cloud_controller.admin_read_only</td>
<td>This admin scope gives read permissions to Cloud Controller.</td>
</tr>
<tr>
<td>cloud_controller.global_audit</td>
<td>This scope gives read-only access to all Cloud Controller API resources except for secrets such as environment variables.</td>
</tr>
</tbody>
</table>

Routing Scopes

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>routing.routes.read</td>
<td>This scope gives the ability to read the full routing table from the router.</td>
</tr>
<tr>
<td>routing.routes.write</td>
<td>This scope gives the ability to write the full routing table from the router.</td>
</tr>
<tr>
<td>routing.router_groups.read</td>
<td>This scope gives the ability to read the full list of routing groups.</td>
</tr>
<tr>
<td>routing.router_groups.write</td>
<td>This scope gives the ability to write the full list of routing groups.</td>
</tr>
</tbody>
</table>

Other Scopes

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doppler.firehose</td>
<td>This scope gives the ability to read logs from the Loggregator Firehose endpoint.</td>
</tr>
<tr>
<td>notifications.write</td>
<td>This scope gives the ability to send notifications through the Notification Service.</td>
</tr>
</tbody>
</table>
This topic describes Garden, the component that Cloud Foundry uses to create and manage isolated environments called containers. Each instance of an application deployed to Cloud Foundry runs within a container. For more information about how containers work, see the Container Mechanics section of the Container Security topic.

Backends

Garden has pluggable backends for different platforms and runtimes, and specifies a set of interfaces that each platform-specific backend must implement. These interfaces contain methods to perform the following actions:

- Create and delete containers
- Apply resource limits to containers
- Open and attach network ports to containers
- Copy files into and out of containers
- Run processes within containers
- Stream STDOUT and STDERR data out of containers
- Annotate containers with arbitrary metadata
- Snapshot containers for redeploys without downtime

For more information, see the Garden repository on GitHub.

Garden-runC

Cloud Foundry currently uses the Garden-runC backend, a Linux-specific implementation of the Garden interface using the Open Container Interface (OCI) standard. Previous versions of Cloud Foundry used the Garden-Linux backend.

Note: PAS versions v1.8.8 and above use Garden-runC instead of Garden-Linux.

Garden-runC has the following features:

- Uses the same OCI low-level container execution code as Docker and Kubernetes, so container images run identically across all three platforms
- AppArmor is configured and enforced by default for all unprivileged containers
- Seccomp whitelisting restricts the set of system calls a container can access, reducing the risk of container breakout
- Allows pluggable networking and rootfs management

For more information, see the Garden-runC repository on GitHub.

Garden RootFS (GrootFS)

Garden manages container filesystems through a plugin interface. Cloud Foundry uses the GrootFS plugin for this task. GrootFS is a Linux-specific implementation of the Garden volume plugin interface.

GrootFS performs the following actions:

- Creates container filesystems based on buildpacks and droplets
- Creates container filesystems based on remote docker images
- Authenticates with remote registries when using remote images
- Properly maps UID/GID for all files inside an image
- Executes garbage collection to remove unused volumes
- Applies per container disk quotas
- Provides per container disk usage stats
For more information, see GrootFS Disk Usage and the GrootFS repository on GitHub.
HTTP Routing

This topic describes how the Gorouter, the main component in Cloud Foundry’s routing tier, routes HTTP traffic within Cloud Foundry (CF).

HTTP Headers

Header Size Limit
Gorouter has a limit of 1MB for HTTP Headers.

X-Forwarded-Proto

Header gives the scheme of the HTTP request from the client. In requests forwarded to backends, Gorouter sets the scheme to HTTP if the client made an insecure request (on port 80) or HTTPS if the client made a secure request (on port 443). Developers can configure their apps to reject insecure requests by inspecting the HTTP headers of incoming traffic and rejecting traffic that includes X-Forwarded-Proto with the scheme of HTTP.

X-Forwarded-For

If X-Forwarded-For is present, the Gorouter appends the load balancer’s IP address to it and forwards the list. If X-Forwarded-For is not present, then the Gorouter sets it to the IP address of the load balancer in the forwarded request (some load balancers masquerade the client IP). If a load balancer sends the client IP using the PROXY protocol, then the Gorouter uses the client IP address to set X-Forwarded-For.

If your load balancer terminates TLS on the client side of the Gorouter, it must append these headers to requests forwarded to the Gorouter. For more information, see the Securing Traffic into Cloud Foundry topic.

HTTP Headers for Zipkin Tracing

Zipkin is a tracing system that enables app developers to troubleshoot failures or latency issues. Zipkin provides the ability to trace requests and responses across distributed systems. See Zipkin.io for more information.

When the Zipkin feature is enabled in Cloud Foundry, the Gorouter examines the HTTP request headers and performs the following:

- If the X-B3-TraceId and X-B3-SpanId HTTP headers are not present in the request, the Gorouter generates values for these and inserts the headers into the request forwarded to an application. These values are also found in the Gorouter access log message for the request: x_b3_traceid and x_b3_spanid.
- If the X-B3-TraceId and X-B3-SpanId HTTP headers are present in the request, the Gorouter forwards them unmodified. In addition to these trace and span ids, the Gorouter access log message for the request includes x_b3_parentspanid.

Developers can then add Zipkin trace IDs to their application logging in order to trace app requests and responses in Cloud Foundry.

After adding Zipkin HTTP headers to app logs, developers can use cf logs myapp to correlate the trace and span ids logged by the Gorouter with the trace ids logged by their app. To correlate trace IDs for a request through multiple apps, each app must forward appropriate values for the headers with requests to other applications.

HTTP Headers for App Instance Routing

Developers who want to obtain debug data for a specific instance of an app can use the HTTP header X-CF-APP-INSTANCE to make a request to an app instance.

Perform the following steps to make an HTTP request to a specific app instance:

1. Obtain the GUID of your app:
2. List your app instances and retrieve the index number of the instance you want to debug:

```
$ cf app YOUR-APP --guid
```

3. Make a request to the app route using the HTTP header `X-CF-APP-INSTANCE` set to the concatenated values of the app GUID and the instance index:

```
$ curl app.example.com -H "X-CF-APP-INSTANCE":"YOUR-APP-GUID:YOUR-INSTANCE-INDEX"
```

Forward Client Certificate to Applications

Applications that require mutual TLS (mTLS) need metadata from client certificates to authorize requests. Cloud Foundry supports this use case without bypassing layer-7 load balancers and the Gorouter.

The HTTP header `X-Forwarded-Client-Cert` (XFCC) may be used to pass the originating client certificate along the data path to the application. Each component in the data path must trust that the back end component has not allowed the header to be tampered with.

If you configure the load balancer to terminate TLS and set the XFCC header from the received client certificate, then you must also configure the load balancer to strip this header if it is present in client requests. This configuration is required to prevent spoofing of the client certificate.

The following sections describe supported deployment configurations.

Terminating TLS at Load Balancer

By default, Gorouter forwards arbitrary headers that are not otherwise mentioned in the docs; this includes the XFCC header.

For applications to receive the XFCC header, configure your load balancer to set the XFCC header with the contents of the client certificate received in the TLS handshake.

This mode is enabled when the **TLS terminated for the first time at infrastructure load balancer** option is selected in the Networking configuration screen of the PAS tile.

Terminating TLS at HAProxy

This option allows you to configure support for the XFCC header while leveraging HAProxy. When selected, HAProxy sets the XFCC header to the contents of the client certificate received in the TLS handshake.

Selecting this configuration requires that the load balancer in front of HAProxy is configured to pass through the TLS handshake to HAProxy via TCP.

This mode is enabled when the **TLS terminated for the first time at HAProxy** option is selected in the Networking configuration screen of the PAS tile.

HAProxy trusts the Diego intermediate certificate authority. This trust is enabled automatically and permits mutual authentication between applications that are running on Pivotal Cloud Foundry.

Terminating TLS at Gorouter

If the Gorouter is the first component to terminate TLS, such that it receives the certificate of the originating client in the mutual TLS handshake, the operator should select this option. When selected, Gorouter sets the XFCC header to the contents of the client certificate received in the TLS handshake and strips the XFCC header when present in a request.

Selecting this configuration requires that the load balancer in front of Gorouter is configured to pass through TLS handshake to Gorouter via TCP.

This mode is enabled when the **TLS terminated for the first time at the Router** option is selected in the Networking configuration screen of the PAS tile.

Gorouter trusts the Diego intermediate certificate authority. This trust is enabled automatically and permits mutual authentication between applications that are running on Pivotal Cloud Foundry.
Client-Side TLS

Depending on your needs, you can configure your deployment to terminate TLS at the Gorouter, at the Gorouter and the load balancer, or at the load balancer only. For more information, see the Securing Traffic into Cloud Foundry topic.

TLS to Apps and Other Back End Services

The Gorouter supports TLS and mutual authentication to back end destinations, including app instances, platform services, and any other routable endpoints.

This is enabled by following the steps in Configure Validation of App Instance Identity with TLS.

Preventing Misrouting

As CF manages and balances apps, the internal IP address and ports for app instances change. To keep the Gorouter’s routing tables current, a Route-Emitter on each Diego cell sends a periodic update to all Gorouters through NATS, reminding them of the location of all app instances; the default frequency of these updates is 20 seconds. The Gorouter tracks a time-to-live (TTL) for each route to back end mapping; this TTL defaults to 120 seconds and is reset when the Gorouter receives an updated registration message.

Network partitions or NATS failures can cause the Gorouter’s routing table to fall out of sync, as CF continues to re-create containers across hosts to keep apps running. This can lead to routing of requests to incorrect destinations.

You can configure the Gorouter to handle this problem in two ways:

<table>
<thead>
<tr>
<th>Consistency mode</th>
<th>Security of traffic between Gorouter and Containers</th>
<th>Gorouter Route Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>With TLS Enabled</td>
<td>Encrypted via TLS</td>
<td>Routes are pruned on failure of TLS handshake only. See Gorouter TLS pruning behavior.</td>
</tr>
<tr>
<td>Without TLS Enabled</td>
<td>Plain text</td>
<td>Routes are pruned on TTL expiry</td>
</tr>
</tbody>
</table>

Both of these consistency modes are described below.

With TLS Enabled

This consistency mode is newer and has the following benefits:

- Improved availability for applications by keeping routes in the Gorouter’s routing table when TTL expires
- Increased guarantees against misrouting by validating the identity of backends before forwarding requests
- Increased security by encrypting data in flight from the Gorouter to backends

⚠️ warning: TLS routing requires an additional 32 MB of RAM capacity on your Diego cells per app instance, as well as additional CPU capacity. Check the total amount of Diego cell memory available to allocate in your environment, and if it is less than 32 MB times the number of running app instances, scale out your Diego cells.

⚠️ warning: You may see an increase of memory and CPU usage for your Gorouters after enabling TLS routing. Check the total amount of memory and CPU usage of the Gorouters in your environment, and if they are close to the size limit, consider scaling out your Gorouters before enabling TLS routing.

In this mode, the Diego Route-Emitters send a modified route registration message to NATS that includes a unique identifier for the app instance, as well instructions to use TLS when communicating with the instance. See TLS to Apps and Other Back End Services for details.

Before forwarding traffic to an app instance, the Gorouter initiates a TLS handshake with an Envoy proxy running in each app container. In the TLS handshake, the Envoy proxy presents a certificate generated by Diego for each container which uniquely identifies the container using the same app instance identifier sent by the Route-Emitter, configured in the certificate as a domain Subject Alternative Name (SAN).

If the Gorouter confirms that the app instance identifier in the certificate matches the one received in the route registration message, the Gorouter forwards the HTTP request over the TLS session, and the Envoy proxy then forwards it to the app process. If the instance identifiers do not match, the
Gorouter removes the app instance from its routing table and transparently retries another instance of the app.

Configure Validation of App Instance Identity with TLS

Verifying app identity using TLS improves resiliency and consistency for app routes. The Application Containers configuration pane of the PAS tile includes the following options:

- **Router uses TLS to verify application identity**
  - Enables the Gorouter to verify app identity using TLS. This is the default option.
- **Router and applications use mutual TLS to verify each other's identity**
  - Enables the Gorouter and your apps to verify each other's identity using TLS.

**Without TLS Enabled**

In this consistency mode, the Diego Route-Emitters on each cell send route registration messages that include instructions for the Gorouter to send unencrypted requests to the app instance. If the Gorouter does not receive an update for the route within the time-to-live (TTL) interval, the route is pruned from the Gorouter's routing table. See TLS to Apps and Other Back End Services for details.

This pruning method was developed before the alternative was available.

**Consistency Mode Can Differ by Instance Group**

The Gorouter can validate app instance identity using TLS only when Diego cells are configured appropriately. Because cells are configured for TLS through the instance group that they belong to, the Gorouter can run in different consistency modes with cells in different instance groups. For example, the Gorouter can communicate over TLS and validate the cells in one Isolation Segment, while communicating with cells in another Isolation Segment via plain text and without validating instance identity.

Currently, only Linux cells support the Gorouter validating app instance identities using TLS. With Windows cells, the Gorouter runs without TLS enabled, forwarding requests to Windows apps over plain text and pruning based on route TTL.

**Round-Robin Load Balancing**

The Gorouter uses the round-robin algorithm for load balancing incoming requests to application instances. The Gorouter maintains a dynamically updated list of application instances for each route, and forwards each request for a given route to the next application instance in the list.

**WebSockets**

WebSockets is a protocol providing bi-directional communication over a single, long-lived TCP connection, commonly implemented by web clients and servers. WebSockets are initiated through HTTP as an upgrade request. The Gorouter supports this upgrade handshake, and holds the TCP connection open with the selected application instance. To support WebSockets, the operator must configure the load balancer correctly. Depending on the configuration, clients may have to use a different port for WebSocket connections, such as port 4443, or a different domain name. For more information, see the Supporting WebSockets topic.

**Session Affinity**

The Gorouter supports session affinity, or sticky sessions, for incoming HTTP requests to compatible apps.
With sticky sessions, when multiple instances of an app are running on CF, requests from a particular client always reach the same app instance. This allows apps to store session data specific to a user session.

- To support sticky sessions, configure your app to return a JSESSIONID cookie in responses. The app generates a JSESSIONID as a long hash in the following format:

  1A530637289A03B07199A44E8D531427

- If an app returns a JSESSIONID cookie to a client request, the CF routing tier generates a unique VCAP_ID for the app instance based on its GUID in the following format:

  323f211e-fea3-4161-9bd1-615392327913

- On subsequent requests, the client must provide both the JSESSIONID and VCAP_ID cookies.

  The CF routing tier uses the VCAP_ID cookie to forward client requests to the same app instance every time. The JSESSIONID cookie is forwarded to the app instance to enable session continuity. If the app instance identified by the VCAP_ID crashes, the Gorouter attempts to route the request to a different instance of the app. If the Gorouter finds a healthy instance of the app, it initiates a new sticky session.

  **Note:** CF does not persist or replicate HTTP session data across app instances. If an app instance crashes or is stopped, session data for that instance is lost. If you require session data to persist across crashed or stopped instances, or to be shared by all instances of an app, store session data in a CF marketplace service that offers data persistence.

### Keepalive Connections

#### From Front End Clients

The Gorouter supports keepalive connections from clients and does not close the TCP connection with clients immediately after returning an HTTP response. Clients are responsible for closing these connections.

#### To Back End Servers

If keepalive connections are disabled, the Gorouter closes the TCP connection with an app instance or system component after receiving an HTTP response.

If keepalive connections are enabled, the Gorouter maintains established TCP connections to back ends. The Gorouter supports up to 100 idle connections to each back end:

- If an idle connection exists for a given back end, the Gorouter reuses it to route subsequent requests.
- If no idle connection exists for this back end, the Gorouter creates a new connection.

### Transparent Retries

If the Gorouter cannot establish a TCP connection with a selected application instance, the Gorouter considers the instance ineligible for requests for 30 seconds and transparently attempts to connect to another application instance. Once the Gorouter has established a TCP connection with an application instance, the Gorouter forwards the HTTP request.

When you deploy an app that requires Diego cells to restart or recreate, the app may not respond to a Gorouter request before the keepalive connection breaks. The following table describes how the Gorouter behaves if it cannot establish a TCP connection to an app:

<table>
<thead>
<tr>
<th>If the Gorouter…</th>
<th>and the backend…</th>
<th>then the Gorouter…</th>
</tr>
</thead>
<tbody>
<tr>
<td>cannot establish a TCP connection to a backend</td>
<td>n/a</td>
<td>retries another backend, no more than 3 times</td>
</tr>
<tr>
<td>establishes a TCP connection to a backend and forwards the request</td>
<td>does not respond</td>
<td>waits 15 minutes for a response, and if it errors, does not retry another backend</td>
</tr>
<tr>
<td>establishes a TCP connection to a backend and forwards the request</td>
<td>returns a TCP connection error</td>
<td>returns an error to the client, marks backend ineligible, and does not retry another backend</td>
</tr>
</tbody>
</table>
In all cases, if the app still does not respond to the request, the Gorouter returns a 502 error.
Diego Components and Architecture

Page last updated:

This topic provides an overview of the structure and components of Diego, the container management system for Pivotal Cloud Foundry versions 1.6 and newer.

This topic includes the following sections:

- Architecture Diagram
- Diego Components
  - Diego Brain
  - Diego Cell
  - Database VMs
  - BOSH DNS
- Platform-Specific Components
  - Garden Back Ends
  - App Lifecycle Binaries

Architecture Diagram

Cloud Foundry uses the Diego architecture to manage app containers. Diego components assume app scheduling and management responsibility from the Cloud Controller.

See the following diagram and descriptions for information about the way Diego handles app requests.
1. The Cloud Controller passes requests to stage and run apps to several components on the Diego Brain.

2. The Diego Brain components translate staging and running requests into Tasks and Long Running Processes (LRPs), then submit these to the Bulletin Board System (BBS) through an API over HTTP.

3. The BBS submits the Tasks and LRPs to the Auctioneer, part of the Diego Brain.

4. The Auctioneer distributes these Tasks and LRPs to Cells through an Auction. The Diego Brain communicates with Diego Cells using SSL/TLS protocol.

5. Once the Auctioneer assigns a Task or LRP to a Cell, an in-process Executor creates a Garden container in the Cell. The Task or LRP runs in the container.

6. The BBS tracks desired LRPs, running LRP instances, and in-flight Tasks. It also periodically analyzes this information and corrects discrepancies to ensure consistency between ActualLRP and DesiredLRP counts.

7. The Metron Agent, part of the Cell, forwards app logs, errors, and metrics to the Cloud Foundry Loggregator. For more information, see the Application Logging in Cloud Foundry topic.

**Diego Components**

Diego components run and monitor Tasks and LRPs.

**Diego Brain**

Diego Brain components distribute Tasks and LRPs to Diego Cells, and correct discrepancies between ActualLRP and DesiredLRP counts to ensure fault-tolerance and long-term consistency.

The Diego Brain consists of the following:

- Auctioneer
- CC-Uploader
- File Server
- SSH Proxy
- TPS Watcher
- TCP Route-emitter
- Nsync
- Stager

**Auctioneer**

- Uses the auction package to run Diego Auctions for Tasks and LRPs
- Communicates with Cell Reps over SSL/TLS
- Maintains a lock in the BBS that restricts auctions to one Auctioneer at a time

See the Auctioneer repository on GitHub for more information.

**CC-Uploader**

- Mediates uploads from the Executor to the Cloud Controller
- Translates simple HTTP POST requests from the Executor into complex multipart-form uploads for the Cloud Controller

See the CC-Uploader repository on GitHub for more information.

**File Server**

[Copyright Pivotal Software Inc, 2013-2019]
This “blobstore” serves static assets that can include general-purpose App Lifecycle binaries and app-specific droplets and build artifacts.

See the [File Server repository](https://github.com) on GitHub for more information.

**SSH Proxy**
- Brokers connections between SSH clients and SSH servers running inside instance containers

See [Application SSH](https://github.com), [Application SSH Overview](https://github.com), or the [Diego SSH repository](https://github.com) on GitHub for more information.

**TPS Watcher**
- Provides the Cloud Controller with information about currently running LRPs to respond to `cf apps` and `cf app APP_NAME` requests
- Monitors `ActualLRP` activity for crashes and reports them the Cloud Controller

See the [TPS repository](https://github.com) on GitHub for more information.

**TCP Route-Emitter**
- Monitors `ActualLRP` and `ActualLRP` states, emitting TCP route registration and unregistration messages to the Cloud Foundry routing API when it detects changes
- Periodically emits TCP routes to the Cloud Foundry routing API

**Nsync**
- Listens for app requests to update the `DesiredLRPs` count and updates `DesiredLRPs` through the BBS
- Periodically polls the Cloud Controller for each app to ensure that Diego maintains accurate `DesiredLRPs` counts

See the [Nsync repository](https://github.com) on GitHub for more information.

**Stager**
- Translates staging requests from the Cloud Controller into generic Tasks and LRPs
- Sends a response to the Cloud Controller when a Task completes

See the [Stager repository](https://github.com) on GitHub for more information.

**Diego Cell**

Diego Cell components manage and maintain Tasks and LRPs.

The Diego Cell consists of the following:

- **Rep**
- **Executor**
- **Garden**
- **Metron Agent**
- **Route-Emitter**

**Rep**
- Represents a Cell in Diego Auctions for Tasks and LRPs
- Mediates all communication between the Cell and the BBS
- Ensures synchronization between the set of Tasks and LRPs in the BBS with the containers present on the Cell
- Maintains the presence of the Cell in the BBS
- Runs Tasks and LRPCs by asking the in-process Executor to create a container and RunAction recipes

See the [Rep repository](https://github.com) on GitHub for more information.

**Executor**

- Runs as a logical process inside the Rep
- Implements the generic Executor actions detailed in the [API documentation](https://github.com)
- Streams STDOUT and STDERR to the Metron agent running on the Cell

See the [Executor repository](https://github.com) on GitHub for more information.

**Garden**

- Provides a platform-independent server and clients to manage Garden containers
- Defines the Garden-runC interface for container implementation

See the [Garden topic](https://github.com) or the [Garden repository](https://github.com) on GitHub for more information.

**Metron Agent**

- Forwards app logs, errors, and app and Diego metrics to the Loggregator Doppler component

See the [Metron repository](https://github.com) on GitHub for more information.

**Route-Emitter**

- Monitors DesiredLRP and ActualLRP states, emitting route registration and unregistration messages to the Cloud Foundry Gorouter when it detects changes
- Periodically emits the entire routing table to the Cloud Foundry Gorouter

See the [Route-Emitter repository](https://github.com) on GitHub for more information.

**Database VMs**

The Diego database VM consists of the following components.

**Diego Bulletin Board System**

- Maintains a real-time representation of the state of the Diego cluster, including all desired LRPCs, running LRPC instances, and in-flight Tasks
- Provides an RPC-style API over HTTP to Diego Core components and external clients, including the SSH Proxy and Route Emitter
- Ensure consistency and fault tolerance for Tasks and LRPCs by comparing desired state (stored in the database) with actual state (from running instances)
- Acts to keep DesiredLRP count and ActualLRP count synchronized in the following ways:
  - If the DesiredLRP count exceeds the ActualLRP count, requests a start auction from the Auctioneer
  - If the ActualLRP count exceeds the DesiredLRP count, sends a stop message to the Rep on the Cell hosting an instance
- Monitors for potentially missed messages, resending them if necessary

See the [Bulletin Board System repository](https://github.com) on GitHub for more information.

**MySQL**

- Provides a consistent key-value data store to Diego
Locket

- Provides a consistent key-value store for maintenance of distributed locks and component presence

Go MySQL Driver

The Diego BBS stores data in MySQL. Diego uses the Go MySQL Driver to communicate with MySQL.

See the Go MySQL Driver repository on GitHub for more information.

BOSH DNS

- Provides service discovery through colocated DNS servers on all BOSH-deployed VMs
- Provides client-side load-balancing by randomly selecting a healthy VM when multiple VMs are available

See the BOSH DNS documentation for more information.

Platform-Specific Components

Garden Back Ends

Garden contains a set of interfaces that each platform-specific back end must implement. See the Garden topic or the Garden repository on GitHub for more information.

App Lifecycle Binaries

The following three platform-specific binaries deploy apps and govern their lifecycle:

- The **Builder**, which stages a CF app. The Builder runs as a Task on every staging request. It performs static analysis on the app code and does any necessary pre-processing before the app is first run.
- The **Launcher**, which runs a CF app. The Launcher is set as the Action on the DesiredLRP for the app. It executes the start command with the correct system context, including working directory and environment variables.
- The **Healthcheck**, which performs a status check on running CF app from inside the container. The Healthcheck is set as the Monitor action on the DesiredLRP for the app.

Current Implementations

- **Buildpack App Lifecycle** implements the Cloud Foundry buildpack-based deployment strategy.
- **Docker App Lifecycle** implements a Docker deployment strategy.
Application SSH Components and Processes

Page last updated:

This document describes details about the Pivotal Application Service (PAS) SSH components for access to deployed application instances. Pivotal Application Service (PAS) supports native SSH access to applications and load balancing of SSH sessions with the load balancer for your PAS deployment.

The SSH Overview document describes procedural and configuration information about application SSH access.

SSH Components

The PAS SSH includes the following central components, which are described in more detail below:

- An implementation of an SSH proxy server.
- A lightweight SSH daemon.

If these components are deployed and configured correctly, they provide a simple and scalable way to access containers apps and other long running processes (LRPs).

SSH Daemon

The SSH daemon is a lightweight implementation that is built around the Go SSH library. It supports command execution, interactive shells, local port forwarding, and secure copy. The daemon is self-contained and has no dependencies on the container root file system.

The daemon is focused on delivering basic access to application instances in PAS. It is intended to run as an unprivileged process, and interactive shells and commands will run as the daemon user. The daemon only supports one authorized key, and it is not intended to support multiple users.

The daemon can be made available on a file server and Diego LRPs that want to use it can include a download action to acquire the binary and a run action to start it. PAS applications will download the daemon as part of the lifecycle bundle.

SSH Proxy Authentication

The SSH proxy hosts the user-accessible SSH endpoint and is responsible for authentication, policy enforcement, and access controls in the context of PAS. After a user has successfully authenticated with the proxy, the proxy will attempt to locate the target container and create an SSH session to a daemon running inside the container. After both sessions have been established, the proxy will manage the communication between the user's SSH client and the container's SSH Daemon.
How Diego Balances App Processes

Diego balances app processes over the virtual machines (VMs) in a Cloud Foundry (CF) installation using the Diego Auction. When new processes need to be allocated to VMs, the Diego Auction determines which ones should run on which machines. The auction algorithm balances the load on VMs and optimizes app availability and resilience. This topic explains how the Diego Auction works.

Refer to the [Auction repository](https://github.com) on GitHub for source code and more information.

Tasks and Long-Running Processes

The Diego Auction distinguishes between two types of jobs: Tasks and Long-Running Processes (LRPs).

- **Tasks** run once, for a finite amount of time. A common example is a staging task that compiles an app’s dependencies, to form a self-contained droplet that makes the app portable and runnable on multiple VMs. Other examples of tasks include making a database schema change, bulk importing data to initialize a database, and setting up a connected service.

- **Long-Running Processes** run continuously, for an indefinite amount of time. LRPs terminate only if stopped or killed, or if they crash. Examples include web servers, asynchronous background workers, and other applications and services that continuously accept and process input. To make high-demand LRPs more available, Diego may allocate multiple instances of the same application to run simultaneously on different VMs, often spread across Availability Zones that serve users in different geographic regions.

The Diego Auction process repeats whenever new jobs need to be allocated to VMs. Each auction distributes a current batch of work, Tasks and LRPs, that can include newly-created jobs, jobs left unallocated in the previous auction, and jobs left orphaned by failed VMs. Diego does not redistribute jobs that are already running on VMs. Only one auction can take place at a time, which prevents placement collisions.

Ordering the Auction Batch

The Diego Auction algorithm allocates jobs to VMs to fulfill the following outcomes, in decreasing priority order:

1. Keep at least one instance of each LRP running.
2. Run all of the Tasks in the current batch.
3. Distribute as much of the total desired LRP load as possible over the remaining available VMs, by spreading multiple LRP instances broadly across VMs and their Availability Zones.

To achieve these outcomes, each auction begins with the Diego Auctioneer component arranging the batch’s jobs into a priority order. Some of these jobs may be duplicate instances of the same process that Diego needs to allocate for high-traffic LRPs, to meet demand. So the Auctioneer creates a list of multiple LRP instances based on the desired instance count configured for each process.

For example, if the process LRP-A has a desired instance count of 3 and a memory load of 2, and process LRP-B has 2 desired instances and a load of 5, the Auctioneer creates a list of jobs for each process as follows:

<table>
<thead>
<tr>
<th>Process</th>
<th>Desired Instances</th>
<th>Load</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRP-A</td>
<td>3</td>
<td>2</td>
<td><img src="image" alt="LRP instances" /></td>
</tr>
<tr>
<td>LRP-B</td>
<td>2</td>
<td>5</td>
<td><img src="image" alt="LRP instances" /></td>
</tr>
</tbody>
</table>

The Auctioneer then builds an ordered sequence of LRP instances by cycling through the list of LRPs in decreasing order of load. With each cycle, it adds another instance of each LRP to the sequence, until all desired instances of the LRP have been added. With the example above, the Auctioneer would order the LRPs like this:
The Auctioneer then builds an ordered sequence for all jobs, both LRPs and Tasks. Reflecting the auction batch priority order, the first instances of LRPs are first priority. Tasks are next, in decreasing order of load. Duplicate LRP jobs come last.

Adding one-time Task-C (load = 4) and Task-D (load = 3) to the above example, the priority order becomes:

Auctioning the Batch to the Cells

With all jobs sorted in priority order, the Auctioneer allocates each in turn to one of the VMs. The process resembles an auction, where VMs “bid” with their suitability to run each job. Facilitating this process, each app VM has a resident Cell that monitors and allocates the machine’s operation. The Cell participates in the auction on behalf of the virtual machine that it runs on.

Starting with the highest-priority job in the ordered sequence, the Auctioneer polls all the Cells on their fitness to run the currently-auctioned job. Cells “bid” to host each job according to the following priorities, in decreasing order:

1. Allocate all jobs only to Cells that have the correct software stack to host them, and sufficient resources given their allocation so far during this auction.

2. Allocate LRP instances into Availability Zones that are not already hosting other instances of the same LRP.

3. Within each Availability Zone, allocate LRP instances to run on Cells that are not already hosting other instances of the same LRP.

4. Allocate any job to the Cell that has lightest load, from both the current auction and jobs it has been running already. In other words, distribute the total load evenly across all Cells.

Our example auction sequence has seven jobs: five LRP instances and two Tasks. The following diagram shows how the Auctioneer might distribute this work across four Cells running in two Availability Zones:
If the Auctioneer reaches the end of its sequence of jobs, having distributed all jobs to the Cells, it submits requests to the Cells to execute their allotted work. If the Cells ran out of capacity to handle all jobs in the sequence, the Auctioneer carries the unallocated jobs over and merges them into the next auction batch, to be allocated in the next auction.

Triggering Another Auction

The Diego Auction process repeats to adapt a Cloud Foundry deployment to its changing workload. For example, the BBS initiates a new auction when it detects that the actual number of running instances of LRP does not match the number desired. Diego’s BBS component monitors the number of instances of each LRP that are currently running. The BBS component periodically compares this number with the desired number of LRP instances, as configured by the user. If the actual number falls short of what is desired, the BBS triggers a new auction. In the case of a surplus of application instances, the BBS kills the extra instances and initiates another auction.

The Cloud Controller also triggers an auction whenever a Cell fails. After any auction, if a Cell responds to its work request with a message that it cannot perform the work after all, the Auctioneer carries the unallocated work over into the next batch. But if the Cell fails to respond entirely, for example if its connection times out, the unresponsive Cell may still be running its work. In this case, the Auctioneer does not automatically carry the Cell’s work over to the next batch. Instead, the Auctioneer defers to the BBS to continue monitoring the states of the Cells, and to re-assign unassigned work later if needed.
PCF Operator Guide

For PCF Operators

This guide shows you how to run a PCF platform. This ongoing responsibility may include but is not limited to:

- Configuring PCF capabilities
- Integrating PCF with external systems
- Updating PCF and installed products
- Monitoring PCF health and performance
- Adjusting PCF resources and options to fix health or performance issues
- Diagnosing and troubleshooting PCF problems
- Managing Pivotal Application Service (PAS) users, resources, and infrastructure
- Installing software services and otherwise enabling PCF developers
- Maintaining PCF
- Keeping PCF secure

If you do these things, you are a PCF **operator**, and the contents of this guide are for you.

Guide Contents

- **Day 2 Configurations** - Setting up internal operations and external integrations for PCF.
- **Ongoing Operations** - Routine procedures for running and growing the platform, including:
  - PCF Upgrades
  - IaaS Changes
  - Monitoring, Logging, and Reporting
  - Platform Tuning
  - Enabling Developers
  - Backing Up
  - Security
- **Managing PAS Runtimes** - Procedures performed by people with administrator or manager roles in Pivotal Application Service (PAS), such as managing users, orgs, spaces, and service instances. Operators can perform these actions by logging in with Admin credentials, which grants them the role of Org Manager across all PAS orgs.
- **Using Ops Manager** - Ops Manager’s dashboard interface streamlines the installation, configuration, and upgrading of PCF platform services and add-ons.
- **Using the Cloud Foundry Command Line Interface (cf CLI)** - Using the cf CLI to send commands to the Cloud Controller, the executive component of PAS.
- **Troubleshooting and Diagnostics** - Tools and procedures for troubleshooting PCF.
Deploying Service Mesh (Beta)

This topic describes how to deploy service mesh for Pivotal Application Service (PAS).

For more information about service mesh, see Service Mesh (Beta).

Enable in PAS

To deploy service mesh, configure PAS as follows:

1. Navigate to the Networking - Service Mesh pane of the PAS tile.

2. Under Service Mesh (Beta), select Enable.

3. For IP Addresses for Ingress Router, do the following depending on your IaaS:
   - vSphere: Enter static IPs for the Istio Routers. You must configure your load balancer with these IPs as well.
   - Other: Leave this field blank.

4. For Ingress Router TLS Keypairs, complete the following fields. You can add more than one keypair if desired using the Add button.
- **Name**: Enter a name for the keypair.
- **Certificate and Private Key for Istio Router**: Enter the Private key and certificate for TLS handshakes with clients. These must be in PEM block format.

5. Click **Save**.

### Create a Load Balancer

To configure a load balancer for service mesh, do the following. The exact procedure varies by IaaS.

1. Create a load balancer with the following:
   - A static IP
   - Health check port **8002** and path **/healthcheck**
   - Firewall rules to allow the following:
     - HTTP on port **80**
     - HTTP on port **8002**
     - TLS on port **443**

2. Navigate to your DNS provider and create a DNS name, ***.mesh.YOUR-CF-APPS-DOMAIN** , that resolves to the IP of the load balancer. This is the default domain for service mesh and it is not configurable.

### Add Load Balancer to Resource Config

If your deployment is on an IaaS other than vSphere, do the following after you create your load balancer:

1. Navigate to the **Resource Config** pane of the PAS tile.
2. In the **Load Balancer** column of the **istio-router** row, enter the name of the load balancer you created.
3. Click **Apply Changes**.
The diagram below shows the key Pivotal Application Service (PAS) network components.

**Load Balancer**

PAS includes an HAProxy load balancer for terminating SSL. If you do not want to serve SSL certificates for PAS on your own load balancer use the HAProxy. If you do choose to manage SSL yourself, omit the HAProxy by setting the number of instances to zero in Ops Manager.

**Router**

The routers in PAS are responsible for routing HTTP requests from web clients to application instances in a load balanced fashion. The routers are dynamically configured based on users mapping of applications to location URLs called routes, and updated by the runtime service as application instances are dynamically distributed.

For high availability, the routers are designed to be horizontally scalable. Configure your load balancer to distribute incoming traffic across all router instances.

Refer to the Cloud Foundry [Architecture](#) topic for more information about Cloud Foundry components.
Identifying the API Endpoint for Your PAS Instance

Page last updated:

The API endpoint for your Pivotal Application Service (PAS) deployment, its target URL, is the API endpoint of the deployment’s Cloud Controller. Find your Cloud Controller API endpoint by consulting your cloud operator, from the Apps Manager, or from the command line.

From the Apps Manager

Log in to the Apps Manager for your PAS instance, then click Tools in the left navigation panel. The Getting Started section of the Tools page shows your API endpoint.

```
GETTING STARTED

$ cf help
$ cf login -a https://api.your_endpoint.com
API endpoint: https://api.your_endpoint.com
Username: your_username
Password: your_password
Org: your_org
Space: your_space
$ cf push your_app
```

From the Command Line

From a command line, use the `cf api` command to view your API endpoint.

Example:

```
$ cf api
API endpoint: https://api.example.com (API version: 2.2.0)
```
Configuring SSL/TLS Termination at HAProxy

Both Pivotal Application Service (PAS) and Isolation Segments for Pivotal Cloud Foundry include an HAProxy instance.

HAProxy is appropriate to use in a deployment when features are needed that are offered by HAProxy but are not offered by the CF Routers or IaaS-provided load balancers such as with Azure load balancers. These include filtering of protected domains from trusted networks.

While HAProxy instances provide load balancing for the Gorouters, HAProxy is not itself highly available. For production environments, use a highly-available load balancer to scale HAProxy horizontally. The load balancer does not need to terminate TLS or even operate at layer 7 (HTTP); it can simply provide layer 4 load balancing of TCP connections. Use of HAProxy does not remove the need for CF Routers; the Gorouter must always be deployed for HTTP applications, and TCP Router for non-HTTP applications.

You can generate a self-signed certificate for HAProxy if you do not want to obtain a signed certificate from a well-known certificate authority.

Procedure: Terminate SSL/TLS at HAProxy

In PCF, perform the following steps to configure SSL termination on HAProxy:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Application Service tile in the Installation Dashboard.
3. Click Networking.
4. Configure the following based on the IaaS of your PCF deployment.

<table>
<thead>
<tr>
<th>If your PCF deployment is on:</th>
<th>Then configure the following:</th>
<th>See also:</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack or vSphere</td>
<td>Decide whether you want your HAProxy to be highly available.</td>
<td>For more information, see Configuring PAS.</td>
</tr>
<tr>
<td></td>
<td>● If you need highly available HAProxy, then perform the following steps:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Choose an IP address for each HAProxy instance on the subnet where you deployed PCF.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. In the HAProxy IPs field of the Networking page, enter the IP addresses you have selected for your HAProxy instances.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Configure your load balancer (for example, FS or NSX) to forward domain names to the HAProxy IP addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● If you do not require high availability (for example, you are setting up a development environment), then perform the following steps:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Skip setting up the load balancer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Choose one IP address for the single HAProxy instance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Configure DNS to point at the IP address. See How to Set Up DNS for HAProxy.</td>
<td></td>
</tr>
<tr>
<td>AWS, GCP or Azure</td>
<td>1. Leave the HAProxy IP address blank.</td>
<td>For more information, see Configuring PAS.</td>
</tr>
<tr>
<td></td>
<td>2. In the Resource Config page of PAS tile, locate the HAProxy job.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. In the Load Balancer column for the HAProxy job, specify the appropriate IaaS load balancer resource.</td>
<td></td>
</tr>
</tbody>
</table>

5. In the Certificates and Private Keys for HAProxy and Router field, click the Add button to define at least one certificate keypair for HAProxy and Router. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in PAS. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate for PCF Deployments.

6. In the Minimum version of TLS supported by HAProxy and Router field, select the minimum version of TLS to use in HAProxy communications. HAProxy use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the HAProxy, see TLS Cipher Suites.

7. Under HAProxy forwards requests to Router over TLS, leave Enabled selected and provide the backend certificate authority.

8. If you want to use a specific set of TLS ciphers for HAProxy, configure TLS Cipher Suites for HAProxy. Enter an ordered, colon-separated list of TLS
cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. For a list of TLS ciphers supported by the HAProxy, see TLS Cipher Suites.

9. If you expect requests larger than the default maximum of 16 Kbytes, enter a new value (in bytes) for HAProxy Request Max Buffer Size. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers.

10. If you want to force browsers to use HTTPS when making requests to HAProxy, select Enable in the HAProxy support for HSTS field and complete the following optional configuration steps:
   a. (Optional) Enter a Max Age in Seconds for the HSTS request. By default, the age is set to one year. HAProxy will force HTTPS requests from browsers for the duration of this setting.
   b. (Optional) Select the Include Subdomains checkbox to force browsers to use HTTPS requests for all component subdomains.
   c. (Optional) Select the Enable Preload checkbox to force instances of Google Chrome, Firefox, and Safari that access your HAProxy to refer to their built-in lists of known hosts that require HTTPS, of which HAProxy is one. This ensures that the first contact a browser has with your HAProxy is an HTTPS request, even if the browser has not yet received an HSTS header from your HAProxy.

11. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select Disable SSL certificate verification for this environment. Selecting this checkbox also disables SSL verification for route services.

   Use this checkbox only for development and testing environments. Do not select it for production environments.

12. (Optional) If you do not want HAProxy or the Gorouter to accept any non-encrypted HTTP traffic, select the Disable HTTP on HAProxy and Router checkbox.

13. In the Configure the CF Router support for the X-Forwarded-Client-Cert header field, select Always forward the XFCC header in the request, regardless of the whether the client connection is mTLS.

14. (Optional) If your PCF deployment uses HAProxy and you want it to receive traffic only from specific sources, use the following fields:
   - Protected Domains: Enter a comma-separated list of domains from which PCF can receive traffic.
   - Trusted CIDRs: Optionally, enter a space-separated list of CIDRs to limit which IP addresses from the Protected Domains can send traffic to PCF.

15. Click Save.

### How to Set Up DNS for HAProxy

You only need to perform this procedure if you are using one instance of HAProxy such as in a development environment. If you would like HAProxy to be highly available, you must have a load balancer in front of it. In this case, you would point DNS at the load balancer.

To use a single instance HAProxy load balancer in a vSphere or OpenStack deployment, create a wildcard A record in your DNS and configure some fields in the PAS product tile.

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the System Domain and Apps Domain that you configure in the Domains section of the PAS tile with the HAProxy IP address.

   For example, with `cf.example.com` as the main subdomain for your Cloud Foundry (CF) deployment and an HAProxy IP address `203.0.113.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `203.0.113.1`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.cf</td>
<td>A</td>
<td>203.0.113.1</td>
<td>example.com</td>
</tr>
</tbody>
</table>

2. Use the Linux `host` command to test your DNS entry. The `host` command should return your HAProxy IP address.

   Example:

   ```bash
   $ host cf.example.com
   cf.example.com has address 203.0.113.1
   $ host anything.example.com
   anything.cf.example.com has address 203.0.113.1
   ```
Configuring Frontend Idle Timeout for Gorouter and HAProxy

This topic describes how to configure the Frontend Idle Timeout for Gorouter and HAProxy field in the Pivotal Application Service (PAS) Networking pane.

You can optionally use the Frontend Idle Timeout for Gorouter and HAProxy field to help prevent connections from your load balancer to Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

In general, set the value higher than your load balancer’s backend idle timeout to avoid the race condition where the load balancer sends a request before it discovers that Gorouter or HAProxy has closed the connection.

See the following table for specific guidance and exceptions to this rule:

<table>
<thead>
<tr>
<th>IaaS</th>
<th>Guidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>AWS ELB has a default timeout of 60 seconds, so Pivotal recommends a value greater than 60.</td>
</tr>
<tr>
<td>Azure</td>
<td>By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, Pivotal recommends a value lower than 240 to force the load balancer to send the TCP RST.</td>
</tr>
<tr>
<td>GCP</td>
<td>GCP has a default timeout of 600 seconds. For GCP HTTP load balancers, Pivotal recommends a value greater than 600. For GCP TCP load balancers, Pivotal recommends a value less than 600 to force the load balancer to send a TCP RST.</td>
</tr>
<tr>
<td>Other</td>
<td>Set the timeout value to be greater than that of the load balancer’s backend idle timeout.</td>
</tr>
</tbody>
</table>
Configuring Proxy Settings for All Applications

This topic describes how to globally configure proxy settings for all applications in your Pivotal Cloud Foundry (PCF) deployment. Some environments restrict access to the Internet by requiring traffic to pass through an HTTP or HTTPS proxy. PCF operators can use the Cloud Foundry Command Line Interface (cf CLI) to provide the proxy settings to all applications, including system applications and service brokers.

Note: Incorrectly configuring proxy settings can prevent applications from connecting to the Internet or accessing required resources. They can also cause errands to fail and break system applications and service brokers. Although errands, system applications, and service brokers do not need to connect to the Internet, they often need to access other resources on PCF. Incorrect proxy settings can break these connections.

Set Environment Variables

To globally configure proxy settings for PCF applications, perform the following steps to set three environment variables for both the staging environment variable group and the running environment variable group.

For more information about variable groups, see the Environment Variable Groups section in the Cloud Foundry Environment Variables topic.

This procedure explains how to set proxy information for both staging and running applications. However, you can set proxy settings for only staging or only running applications.

1. Target your Cloud Controller with the cf CLI. If you have not installed the cf CLI, see the Installing the cf CLI topic.

   $ cf api YOUR-SYSTEM-DOMAIN
   Setting api endpoint to api.YOUR-SYSTEM-DOMAIN...
   OK
   API endpoint: https://api.YOUR-SYSTEM-DOMAIN (API version: 2.54.0)
   Not logged in. Use `cf login` to log in.

2. Log in with your UAA administrator credentials. To retrieve these credentials, navigate to the Pivotal Application Service (PAS) tile in the Ops Manager Installation Dashboard and click Credentials. Under UAA, click Link to Credential next to Admin Credentials and record the password.

   $ cf login
   API endpoint: https://api.YOUR-SYSTEM-DOMAIN
   Email: admin
   Password:
   Authenticating...
   OK

3. To configure proxy access for applications that are staging, run the following command, replacing the placeholder values:

   - `http_proxy`: Set this value to the proxy to use for HTTP requests.
   - `https_proxy`: Set this value to the proxy to use for HTTPS requests. In most cases, this will be the same as `http_proxy`.
   - `no_proxy`: Set this value to a comma-separated list of DNS names or IP addresses that can be accessed without passing through the proxy. This value may not be needed, because it depends on your proxy configuration. From now on, the proxy settings are applied to staging applications.

4. To configure proxy access for applications that are running, run the following command, replacing the placeholder values as above:


   To configure proxy settings for Java-based applications, use the following command instead, replacing the placeholder values. For `http.nonProxyHosts`, use a pipe-delimited list rather than a comma-separated list.

   $ cf set-running-environment-variable-group '{"JAVA_OPTS": "-Dhttp.proxyHost=YOUR-PROXY -Dhttp.proxyPort=8080 -Dhttp.nonProxyHosts=NO-PROXY.EXAMPLE.COM"}'

   For more information about these Java proxy settings, see Java Networking and Proxies.

5. To apply the proxy configuration for the running environment variable group, you must restart each application that you want to use the new configuration.
Troubleshooting

If an application fails after you apply the global proxy settings, try the following solutions.

Exclude an App From Global Proxy Settings

If your application fails, try instructing the application to ignore the global proxy settings. Perform the following commands to manually unset the proxy environment variables for the failing application:

1. Set the proxy environment variables for `http_proxy` to an empty value:
   
   ```bash
   $ cf set-env YOUR-APP http_proxy ''
   ```

2. Set the proxy environment variables for `https_proxy` to an empty value:
   
   ```bash
   $ cf set-env YOUR-APP https_proxy ''
   ```

3. Set the proxy environment variables for `no_proxy` to an empty value:
   
   ```bash
   $ cf set-env YOUR-APP no_proxy ''
   ```

Change Case of HTTP

Your application and language runtime may be case-sensitive. Try performing the steps in the Set Environment Variables section using uppercase for `HTTP_PROXY`, `HTTPS_PROXY`, and `NO_PROXY` instead of lowercase. Refer to the following example.

```bash
```

Check Proxy Settings

If you have set up your proxy so that it can only send traffic to the Internet, then a request to an internal resource like PCF fails. You must set `no_proxy` so that traffic destined for PCF and other internal resources is sent directly and does not go through the proxy. For instance, setting `no_proxy` to include your system and application domains will ensure that requests destined for those domains are sent directly.

Verify Interpretation

The interpretation of `no_proxy` depends on the application and the language runtime. Most support `no_proxy`, but the specific implementation may vary. For example, some match DNS names that end with the value set in `no_proxy`: `example.com` would match `test.example.com`. Others support the use of the asterisk as a wildcard to provide basic pattern matching in DNS names: `*.example.com` would match `test.example.com`. Most applications and language runtimes do not support pattern matching and wildcards for IP addresses.
Restricting App Access to Internal PCF Components

This topic describes how to secure the component virtual machines (VMs) of your Pivotal Cloud Foundry (PCF) deployment from being accessed by apps.

Introduction

See the following list to understand the concepts for this topic:

- **How PCF determines where apps can send traffic:**
  - PCF uses Application Security Groups (ASGs), which are network policy rules specifying protocols, ports, and IP ranges that apply to outbound network connections initiated from apps. See ASGs.

- **Why you must create new rules for outbound app traffic:**
  - PCF installs with a default ASG that allows apps running on your deployment to send traffic to almost any IP address. This means apps are not blocked from initiating connections to most network destinations unless an administrator takes action to update the ASGs with a more restrictive policy.

- **How you can set up new rules:**
  - To help secure your component VMs against apps while ensuring your apps can access the services they need, follow the procedure below, which includes these steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Determine Your Network Layout:</strong> The procedure for securing your deployment with ASGs varies depending on your network layout, which you can determine using Ops Manager.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Ensure Access for PCF System Apps:</strong> Bind the default ASG to the <code>system</code> org so that PCF system apps can continue accessing the system components they need after you remove the deployment-wide default ASG in Step 4.</td>
</tr>
<tr>
<td>3</td>
<td><strong>Create New ASGs:</strong> Block apps from sending traffic to system components, but allow them to send traffic to the services they need.</td>
</tr>
<tr>
<td>4</td>
<td><strong>Remove the Default ASG:</strong> After you create and bind new ASGs, you no longer need the deployment-wide default ASG bindings that allow apps to send traffic to any IP.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Restart your Apps:</strong> To apply the ASG changes, you must restart all of the apps in your deployment.</td>
</tr>
</tbody>
</table>

- **When to set up new rules:**
  - Pivotal recommends that you complete this procedure directly after installing PCF, prior to developers pushing apps to the platform. If you complete the procedure after apps have been pushed to the platform, you must restart all the apps in your deployment.

Prerequisites

The procedure below requires that you have the latest release of ASG Creator from the Cloud Foundry incubator repository on Github. See About the ASG Creator Tool.

Procedure

Follow these steps to apply ASGs that prevent apps running on your deployment from accessing internal PCF components.

**Step 1: Determine Your Network Layout**

The procedure for securing your deployment with ASGs varies depending on your network layout, which you can determine by following these steps:

1. Log in to Ops Manager.
2. For each tile, click Assign AZs and Networks and record the selected Network that the tile is installed on.

3. Based on the information you gathered, determine which of the following network layouts you have:

<table>
<thead>
<tr>
<th>Layout Name</th>
<th>Layout Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Network</td>
<td>- One network for Ops Manager and the BOSH Director, Pivotal Application Service (PAS), and services.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot secure your deployment with ASGs if you have this network layout. Because PCF dynamically allocates IPs, they cannot be easily excluded in the case of a single network.</td>
</tr>
<tr>
<td>Two Networks</td>
<td>- One network for Ops Manager and the BOSH Director.</td>
</tr>
<tr>
<td></td>
<td>- One network for PAS and Services.</td>
</tr>
<tr>
<td>Three Networks</td>
<td>- One network for Ops Manager and the BOSH Director.</td>
</tr>
<tr>
<td></td>
<td>- One network for PAS.</td>
</tr>
<tr>
<td></td>
<td>- One network for all services.</td>
</tr>
<tr>
<td>Three or More Networks</td>
<td>- One network for Ops Manager and the BOSH Director.</td>
</tr>
<tr>
<td></td>
<td>- One network for PAS.</td>
</tr>
<tr>
<td></td>
<td>- One network for each service.</td>
</tr>
</tbody>
</table>

4. If your network layout includes two or more networks, continue Step 2: Ensure Access for PCF System Apps.

**Step 2: Ensure Access for PCF System Apps**

Follow these steps to apply the default ASG to the system org. This provides network access to PCF system apps without restrictions, which enables them to continue functioning properly after you perform Step 4: Remove the Deployment-wide Default ASG Binding.

1. Bind the default ASG to the staging set in the system org:

   
   ```
   $ cf bind-staging-security-group default_security_group
   ```

2. Bind the default ASG to the running set in the system org:

   ```
   $ cf bind-running-security-group default_security_group
   ```

**Step 3: Create New ASGs**

Follow these steps to create ASGs that block apps from accessing PCF components and create any additional ASGs that allow apps to access the services they require.

**Part A: Record CIDRs**

Gather the CIDRs for each network in your deployment:

1. From the BOSH Director tile, click Create Networks within the Settings tab.
2. In the Networks section, expand each network in your deployment by clicking its name.
3. Record the CIDR for each network.

**Part B: Create and Bind ASGs that Block Network Access**

Create ASGs that block apps from sending traffic to the networks that host Ops Manager, PAS, and, optionally, any installed services.
1. Create a `config.yml` containing the appropriate content for your network layout and replace the indicated values with the CIDRs you gathered:

   - **Two Network Layout**:
     ```yaml
     exclude:
     - YOUR-OPS-MANAGER-CIDR
     - YOUR-PAS-AND-SERVICES-CIDR
     ```

   - **Three Network Layout**:
     ```yaml
     exclude:
     - YOUR-OPS-MANAGER-CIDR
     - YOUR-PAS-CIDR
     - YOUR-SERVICES-CIDR
     ```

   - **Three or More Network Layout**:
     ```yaml
     exclude:
     - YOUR-OPS-MANAGER-CIDR
     - YOUR-PAS-CIDR
     - YOUR-SERVICE-CIDR-1
     - YOUR-SERVICE-CIDR-2
     etc...
     ```

2. Run the following command to generate the default `public-networks.json` and `private-networks.json` files that contain your ASG rules, specifying the location of the `config.yml` file as input:

   ```bash
   $ asg-creator create --config config.yml
   ```

3. Create the `public-networks` ASG by running the following command:

   ```bash
   $ cf create-security-group public-networks public-networks.json
   ```

4. Bind the ASG to the default staging set:

   ```bash
   $ cf bind-staging-security-group public-networks
   ```

5. Bind the ASG to the default running set:

   ```bash
   $ cf bind-running-security-group public-networks
   ```

6. Create the `private-networks` ASG by running the following command:

   ```bash
   $ cf create-security-group private-networks private-networks.json
   ```

7. Bind the ASG to the default staging set:

   ```bash
   $ cf bind-staging-security-group private-networks
   ```

8. Bind the ASG to the default running set:

   ```bash
   $ cf bind-running-security-group private-networks
   ```

   **Note:** You can create and bind additional ASGs by following the procedures in `Create ASGs` and `Bind ASGs`.

**Part C: Create and Bind ASGs for Service Access**

© Copyright Pivotal Software Inc, 2013-2019
Now that you have created ASGs to secure the Ops Man, PAS, and service components, work with developers to create additional ASGs that give apps access to the services they need.

For example, in any space where apps need to access the MySQL for PCF service, follow the steps in Creating Application Security Groups for MySQL.

For more information about creating and binding ASGs, see the following:

- Managing ASGs with the cf CLI
- Typical ASGs

### Step 4: Remove the Default ASG

Now that you have bound new ASGs to determine outbound traffic rules, you no longer need the default ASG bindings that allow apps to send traffic to any IP address.

1. Unbind the default ASG from the staging set:

   ```
   $ cf unbind-staging-security-group default_security_group
   ```

2. Unbind the default ASG from the running set:

   ```
   $ cf unbind-running-security-group default_security_group
   ```

### Step 5: Restart your Apps

To apply the ASG changes, you must restart all of the apps in your deployment. To mitigate app downtime during the restart, Pivotal recommends a blue-green deployment strategy.

**Notes:** You do not need to restart the apps in the system org.

1. Work with developers to restart a few of their apps individually and test that they still work correctly with the new ASGs in place. If an app does not work as expected, you likely must create another ASG that allows the app to send traffic to a service it requires.

   **Note:** To quickly roll back to the original overly-permissive state, you can re-bind the default_security_group ASG to the default-staging and default-running sets. You must then restart your apps to re-apply the original ASGs.

2. Restart the rest of the apps running on your deployment. Optionally, you can use the app-restarter cf CLI plugin to restart all apps in a particular space, org, or deployment.
Configuring Application Security Groups for Email Notifications

To allow the Notifications Service to have network access you need to create Application Security Groups (ASGs).

Note: Without Application Security Groups the service is not usable.

Prerequisite

Review the Getting Started with the Notifications Service topic to ensure you have setup the service.

Configure Network Connections

The Notifications Service is deployed as a suite of applications to the notifications-with-ui space in the system org, and requires the following outbound network connections:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Ports</th>
<th>Protocol</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP_SERVER</td>
<td>587 (default)</td>
<td>tcp (default)</td>
<td>This service is used to send out email notifications</td>
</tr>
<tr>
<td>LOAD_BALANCER IP</td>
<td>80, 443</td>
<td>tcp</td>
<td>This service will access the load balancer</td>
</tr>
<tr>
<td>ASSIGNED_NETWORK</td>
<td>3306</td>
<td>tcp</td>
<td>This service requires access to internal services. ASSIGNED_NETWORK is the CIDR of the network assigned to this service.</td>
</tr>
</tbody>
</table>

Note: The SMTP Server port and protocol are dependent on how you configure your server.

Create a SMTP Server ASG

1. Navigate to the Ops Manager Installation Dashboard and click the Pivotal Application Service tile > Settings tab.

2. Record the information in the Address of SMTP Server and Port of SMTP Server fields.

3. Using the Address of SMTP Server information you obtained in the previous step, find the IP addresses and protocol of your SMTP Server from the service you are using. You might need to contact your service provider for this information.

4. Create a smtp-server.json file. For destination, you must enter the IP address of your SMTP Server.

   ```json
   [
     {
       "protocol": "tcp",
       "destination": "SMTP_SERVER_IPS",
       "ports": ["587"]
     }
   ]
   ```

5. Create a security group called smtp-server:

   ```bash
cf create-security-group smtp-server smtp-server.json
   ```

Create a Load Balancer ASG

Note: If you already have a ASG setup for a Load Balancer, you do not need to perform this step. Review your ASGs to check which groups you have setup.
If you are using the built-in HAProxy as your load balancer, follow this procedure. If you are using an external load balancer, you must obtain your HAProxy IPs from the service you are using.

1. Record the HAProxy IPs in the Pivotal Application Service tile > Settings > Networking tab.

2. Create a `load-balancer-https.json` file. For destination, use the HAProxy IPs you recorded above.

   
   ```
   {
     "protocol": "tcp",
     "destination": "10.68.196.250",
     "ports": ["80", "443"]
   }
   ```

3. Create a security group called `load-balancer-https`:

   ```
   $ cf create-security-group load-balancer-https load-balancer-https.json
   ```

Create an Assigned Network ASG

Note: If you use external services, the IP addresses, ports, and protocols depend on the service.

1. Navigate to the Ops Manager Installation Dashboard > Pivotal Application Service tile > Settings > Assign AZs and Networks section.

2. Navigate to the network selected in the dropdown.

3. Record the BOSH Director tile > Settings tab > Create Networks > CIDR for the network identified in the previous step. Ensure the subnet mask allows the space to access `p-mysql`, `p-rabbitmq`, and `p-redis`.

4. Create a file `assigned-network.json`. For the destination, enter the CIDR you recorded above.

   ```
   {
     "protocol": "tcp",
     "destination": "10.68.0.0/20",
     "ports": ["3306", "5672", "6379"]
   }
   ```

5. Create a security group called `assigned-network`:

   ```
   $ cf create-security-group assigned-network assigned-network.json
   ```

Bind the ASGs

1. Target the `system` org:

   ```
   $ cf target -o system
   ```

2. Create a `notifications-with-ui` space:

   ```
   $ cf create-space notifications-with-ui
   ```

3. Bind the ASGs you created in this topic to the `notifications-with-ui` space:

   ```
   $ cf bind-security-group smtp-server system notifications-with-ui
   $ cf bind-security-group load-balancer-https system notifications-with-ui
   $ cf bind-security-group assigned-network system notifications-with-ui
   ```
Configuring SSH Access for PCF

To help troubleshoot applications hosted by a deployment, Pivotal Cloud Foundry (PCF) supports SSH access into running applications. This document describes how to configure a PCF deployment to allow SSH access to application instances, and how to configure load balancing for those application SSH sessions.

Pivotal Application Service Configuration

This section describes how to configure Pivotal Application Service (PAS) to enable or disable deployment-wide SSH access to application instances. In addition to this deployment-wide configuration, Space Managers have SSH access control over their Space, and Space Developers have SSH access control over their Applications. For details about SSH access permissions, see the Application SSH Overview topic.

To configure PAS SSH access for app instances:

1. Open the PAS tile in Ops Manager.
2. Under the Settings tab, select the Application Containers section.
3. Enable or disable the Allow SSH access to app containers checkbox.
4. Optionally, select Enable SSH when an app is created to enable SSH access for new apps by default in spaces that allow SSH. If you deselect this checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.
SSH Load Balancer Configuration

For IaaSes where load-balancing is available as a service, you should provision a load balancer to balance load across SSH proxy instances. Configure this load balancer to forward incoming TCP traffic on port 2222 to a target pool where you deploy diego_brain instances.

For AWS, Azure, and GCP IaaSes, you configure SSH load balancers in the Resource Config pane. To register SSH proxies with a load balancer, enter your load balancer name in the Load Balancers field in the Diego Brain row.

Ops Manager supports an API-only field. You can configure load balancers in vSphere using this field.
Securing Services Instance Credentials with Runtime CredHub

This topic describes how Pivotal Cloud Foundry (PCF) operators can ensure service instance credentials are securely stored in CredHub.

- What is runtime CredHub?
  - The Pivotal Application Service (PAS) tile includes its own CredHub component, separate from the CredHub component included with the BOSH Director tile. For more information about this centralized credential management component, see the CredHub documentation.

- What is runtime CredHub used for?
  - Runtime CredHub exists to securely store service instance credentials. Previously, PCF could only use the Cloud Controller database for storing these credentials.

- What are service instance credentials?
  - When developers want their app to use a service, such as those provided by the Spring Cloud Services tile for PCF, they must bind their app to an instance of that service. Service bindings include credentials that developers can use to access the service from their app. For more information, see Binding Credentials.

- How can I ensure that service instance credentials are stored in runtime CredHub?
  - You must configure PAS to enable this functionality by following the procedure below. Not all services support the use of runtime CredHub.

- Can I use runtime CredHub to store service instance credentials if some of my services do not support the use of runtime CredHub?
  - PAS supports both services that do and do not use runtime CredHub. Services that do not use runtime CredHub continue to pass their credentials to the Cloud Controller database.

- Can I rotate service instance credentials in runtime CredHub?
  - Runtime CredHub supports credential rotation. For more information, see Rotating Runtime CredHub Encryption Keys.

PCF Services that Use Secure Binding Credentials

The procedures in this document are only effective for services that support storing their instance credentials in runtime CredHub. To learn whether a service supports this feature, see the table below or the documentation for that service.

<table>
<thead>
<tr>
<th>Service</th>
<th>Versions with Secure Binding Credentials</th>
<th>Pivotal Network Listing</th>
</tr>
</thead>
<tbody>
<tr>
<td>CredHub Service Broker for PCF</td>
<td>All</td>
<td>CredHub Service Broker for PCF</td>
</tr>
<tr>
<td>MySQL for PCF v2</td>
<td>v2.3 and later (available to user groups only)</td>
<td>MySQL for PCF v2</td>
</tr>
<tr>
<td>Spring Cloud Services for PCF</td>
<td>v1.5 and later</td>
<td>Spring Cloud Services for PCF</td>
</tr>
<tr>
<td>RabbitMQ for PCF</td>
<td>v1.12 and later</td>
<td>RabbitMQ for PCF</td>
</tr>
<tr>
<td>Redis for PCF</td>
<td>v1.13 and later</td>
<td>Redis for PCF</td>
</tr>
</tbody>
</table>

Prerequisites

Breaking Change: If you opt out of the BOSH DNS feature, your PCF deployment cannot support Secure Service Instance Credentials.

Runtime CredHub allows you to use one or more Hardware Security Modules (HSMs) to store encryption keys. If you wish to use an HSM with CredHub, you must configure the HSM before completing the procedures below. For more information, see Preparing CredHub HSMs for Configuration.

Step 1: Configure the PAS Tile

To configure the PAS tile to support securing service instance credentials in CredHub, you need to:

- In the CredHub pane, provide at least one encryption key. CredHub supports multiple encryption key providers.
- In the Resource Config pane, set the number of CredHub instances to at least one.

CredHub configuration options include:
Internal or external databases
Encryption keys stored internally, externally in a Hardware Security Module (HSM), or both

To configure the PAS tile, follow the instructions in Configuring PAS.

Step 2: Create Application Security Groups

Application Security Groups (ASGs) are network policy rules specifying protocols, ports, and IP ranges that apply to outbound network connections initiated from apps. You must follow the steps below to ensure the ASGs for your deployment allow apps to communicate with the runtime CredHub API.

1. From the PAS tile, click Assign AZs and Networks and record the selected Network where the tile is installed.
2. From the BOSH Director tile, within the Settings tab, click Create Networks.
3. In the Networks section, click the name of the PAS network to expand it.
4. Record the CIDR for the PAS network.
5. Create a file named `runtime-credhub.json` for specifying your ASG rules. Copy the content below into the file. Replace `YOUR-PAS-CIDR` with the CIDR you recorded in the previous step.

```json
[
  {
    "protocol": "tcp",
    "destination": "YOUR-PAS-CIDR",
    "ports": "8844"
  }
]
```
6. Run the following command to create an ASG that allows apps to access the CredHub API:

```
$ cf create-security-group runtime-credhub ~/.workspace/runtime-credhub runtime-credhub.json
```
7. Bind this ASG to your deployment or the specific space in which you want apps to access CredHub. For more information about binding ASGs, see Bind ASGs. Ensure that apps deployed as part of the service tile installation process have access to CredHub in addition to the apps pushed to the platform by developers. For example, the Spring Cloud Services tile deploys the `spring-cloud-broker` app to the `p-spring-cloud-services` space of the `system` org.
8. Restart apps for the ASGs to take effect. Optionally, you can use the `app-restarter cf plugin` cf to restart all apps in a particular space, org, or deployment.

Step 3: Unbind and Rebind Service Instances

For any service instance bindings that existed before runtime CredHub was supported for that service, you must work with your developers to unbind and rebind the service instances to their apps. If you do not unbind and rebind the service, apps continue functioning as normal and fetching credentials from the Cloud Controller database.

Note: This step is not required for bindings created after you installed the new version of the service tile that supports CredHub and you completed the procedures in steps 1 and 2 of this topic.

1. Unbind the service instance from the app:

```
$ cf unbind-service YOUR-APP YOUR-SERVICE-INSTANCE
```
2. Rebind the service instance to the app:

```
$ cf bind-service YOUR-APP YOUR-SERVICE-INSTANCE
```
3. Review the `VCAP_SERVICES` environment variable to verify that the new service instance binding includes CredHub pointers:

```
cf env YOUR-APP
```

See the `VCAP_SERVICES` section of Cloud Foundry Environment Variables for help parsing the output of the `cf-env` command.

4. Restart the app to apply the service instance binding:

```
cf restart YOUR-APP
```

If you run `cf-env` again, you can see the `VCAP_SERVICES` environment variable now contains the credentials for the service instance binding.
Identifying PAS Jobs Using vCenter

Page last updated:

To effectively monitor, control, and manage the virtual machines making up your Pivotal Application Service (PAS) deployment, you may need to identify which VM corresponds to a particular job in PAS. You can find the CID of a particular VM from Pivotal Cloud Foundry (PCF) Operations Manager by navigating to PAS Status.

If you have deployed PAS to VMware vSphere, you can also identify which PAS job corresponds to which VM using the vCenter vSphere client.

Note: The CID shown in Ops Manager is the name of the machine in vCenter.

Identifying PAS Jobs Using vCenter

1. Launch the vSphere client and log in to the vCenter Server system.

2. Select the Inventory > Hosts and Clusters view.

3. Select the Resource Pool containing your PAS deployment.

4. Select the Virtual Machines tab.

5. Right-click the column label heading and check job.
6. The job column displays the PAS job associated with each virtual machine.
Configuring Logging in PAS

This topic describes the types of logs that Pivotal Cloud Foundry (PCF) generates. It also explains how to forward system logs to an external aggregator service, how to scale Loggregator component VMs to keep up with app log volume, and how to manage app traffic logging.

System Logs, App Logs, App Traffic Logs

Pivotal Cloud Foundry (PCF) generates two types of logs, system logs from PCF components and app logs from hosted apps, as differentiated in the table below:

<table>
<thead>
<tr>
<th>Log Type</th>
<th>Originate from</th>
<th>Follow format</th>
<th>Stream from</th>
<th>Can to stream out to (configurable)</th>
<th>Visible to</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Logs</td>
<td>Platform components</td>
<td>Syslog standard</td>
<td>rsyslog agent</td>
<td>Component syslog drain</td>
<td>Operators</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>App Logs</td>
<td>Hosted apps</td>
<td>Format is up to the developer</td>
<td>Firehose¹</td>
<td>External data platform, optionally via nozzles</td>
<td>Developers and Operators</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Optional) With Syslog Adapter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Converted to syslog standard</td>
<td>Syslog Adapter</td>
<td>External syslog drain</td>
<td></td>
</tr>
</tbody>
</table>

¹The Loggregator Firehose also streams component metrics.

App traffic logs are system logs. When app containers communicate, or attempt to communicate, their host cells generate app traffic logs. App traffic logs are system logs, not app logs. These logs come from host cells, not apps, and they carry no information from within the app. App traffic logs only show app communication behavior, as detected from outside by the host cell.

Example Uses of the Log Cache CLI Plugin

To access cached logs with the Log Cache CLI plugin, you must first download and install the plugin.

Download the Log Cache CLI plugin [1].

One you have installed the plugin, the basic command to access cached app logs is:

```
$ cf [OPTIONS] [SOURCE ID/APP]
```

Here are some ways you can use Log Cache to filter app logs:

- ```--start-time```: Displays the start of the cache or the start of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with ```--end-time``` to view logs within a time period.
- ```--end-time```: Displays the end of the cache or the end of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with ```--start-time``` to view logs within a time period.
- ```--json```: Output logs in JSON.
- ```--follow```: Append exported logs to ```stdout````.

For more information on using the Log Cache CLI, see Log Cache CLI: Usage [1].

Example uses of the Log Cache API

The Log Cache API is hosted on PCF, and references your system domain to return responses. The root URL for API calls is [https://log-cache.[YOUR-SYSTEM-DOMAIN]].

The basic call to access and filter cached app logs is:

```
GET [https://log-cache.[YOUR-SYSTEM-DOMAIN]/v1/read] [SOURCE ID]
```
Append the following parameters to the `GET` to customize app logs:

- `start_time`: Displays the start of the cache or the start of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with `end_time` to view logs within a time period.
- `end_time`: Displays the end of the cache or the end of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with `start_time` to view logs within a time period.
- `envelope_types`: Filters by Envelope Type. The available filters are: `LOG`, `COUNTER`, `GAUGE`, `TIMER`, and `EVENT`. Set an envelope type filter to emit logs of only that type. Specify this parameter multiple times to include more types.
- `limit`: Sets a maximum number of envelopes to request. The max limit is 1000. This value defaults to 100.

More API parameters are available to customize retrieved app logs. For more information, see Log Cache: RESTful API Gateway.

Enable Syslog Forwarding

You can configure system logging in PAS to forward log messages from PAS component VMs to an external service. Pivotal recommends forwarding logs to an external service for use in troubleshooting.

Note: The following instructions explain how to configure system logging for PAS component VMs. To forward logs from PCF tiles to an external service, you must also configure system logging in each tile. See the documentation for the given tiles for information about configuring system logging.

To configure system logging in PAS, do the following:

1. In the PAS Settings tab, select the System Logging pane. The following image shows the System Logging pane.
Optionally configure rsyslog to forward platform component logs to an external service. If you do not fill these fields, platform logs will not be forwarded but will remain available on the component VMs and for download via Ops Manager.

1. **Address**: Enter the IP address of the syslog server.
2. **Port**: Enter the port of the syslog server. The default port for a syslog server is 514.
3. **Transport Protocol**: Select a transport protocol for log forwarding.
4. **Encrypt syslog using TLS?**
   - **No**
   - **Yes**
5. **Syslog Drain Buffer Size (# of messages)**: 10000
6. **Include container metrics in SysLog Drains**
7. **Enable Cloud Controller security event logging**
8. **Use TCP for file forwarding local transport**
9. **Don’t Forward Debug Logs**
10. **Custom rsyslog Configuration**

### Note:
The host must be reachable from the PAS network and accept UDP or TCP connections. Ensure the syslog server listens on external interfaces.

### Special Considerations:
- **Include container metrics in Syslog Drains**: Enter either the name or SHA1 fingerprint of the remote peer.
- **TLS CA Certificate**: Enter the TLS CA certificate for the remote server.
- **Syslog Drain Buffer Size**: Enter the number of messages from the Loggregator Agent that the Doppler server can store before it begins to drop messages. See the Loggregator Guide for Cloud Foundry Operators topic for more details.

- **Disable the Include container metrics in Syslog Drains checkbox** to prevent the CF Drain CLI plugin from including app container metrics in syslog drains. This feature is enabled by default.
8. Enable the **Enable Cloud Controller security event logging** checkbox to include security events in the log stream. This feature logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

9. Enable the **Use TCP for file forwarding local transport** checkbox to transmit logs over TCP. This prevents log truncation, but may cause performance issues.

10. Disable the **Don't Forward Debug Logs** checkbox to forward DEBUG syslog messages to an external service. This checkbox is enabled by default.

   **Note:** Some PAS components generate a high volume of DEBUG syslog messages. Enabling the **Don't Forward Debug Logs** checkbox prevents PAS components from forwarding the DEBUG syslog messages to external services. However, PAS still writes the messages to the local disk.

11. For **Custom rsyslog Configuration**, enter a custom syslog rule. For more information about adding custom syslog rules, see [Customizing Syslog Rules](#).

12. Click **Save**.

To configure Ops Manager for system logging, see the **Settings** section in the **Using the Ops Manager Interface** topic.

### Include Container Metrics in Syslog Drains

Developers can monitor container metrics over the syslog protocol using the CF Drain CLI plugin. With the CF Drain CLI plugin, you can use the Cloud Foundry Command Line Interface (cf CLI) tool to set the app container to deliver container metrics to a syslog drain. Developers can then monitor the app container based on those metrics.

In PAS, the **Include container metrics in Syslog Drains** checkbox is enabled by default in the **System Logging** pane. If you have security concerns about streaming container metrics from your app, you can disable this checkbox.

For more information, see [Including Container Metrics in Syslog Drains](#) in **Application Logging in Cloud Foundry**.

### Scale Loggregator

Apps constantly generate app logs and PCF platform components constantly generate component metrics. The Loggregator system combines these data streams and handles them as follows. See [Loggregator Architecture](#) for more information.

- The **Loggregator** agent running on each component or application VM collects and sends this data out to Doppler components.
- Doppler components temporarily buffer the data before periodically forwarding it to the Traffic Controller. When the log and metrics data input to a Doppler exceeds its buffer size for a given interval, data can be lost.
- The **Traffic Controller** serves the aggregated data stream through the Firehose WebSocket endpoint.

Follow the instructions below to scale the Loggregator system. For guidance on monitoring and capacity planning, see [Monitoring Pivotal Cloud Foundry](#).

### Add Component VM Instances

1. From the PAS tile, select **Resource Config**.

2. Increase the number in the **Instances** column of the component you want to scale. You can add instances for the following Loggregator components:

   - **Loggregator Traffic Controller**

     **Note:** The **Reverse Log Proxy (RLP)** BOSH job is colocated on the Traffic Controller VM. If you want to scale Loggregator to handle more logs for syslog drains, you can add instances of the Traffic Controller.

     **Note:** The **BOSH System Metrics Forwarder** job is colocated on the Traffic Controller VM. If you want to scale Loggregator to handle more BOSH component metrics, you can add instances of the Traffic Controller.

   - **Syslog Adapter**
   - **Doppler Server**
3. Click **Save**.

4. Click **Review Pending Changes**, then **Apply Changes**.

---

## App Traffic Logging

App traffic logging generates logs when app containers communicate with each other directly, or attempt to communicate, as allowed by container-to-container networking (C2C) policies and Application Security Groups (ASGs).

App traffic logging lets network security teams audit C2C traffic, by seeing allowed and denied packets, without needing access to the Cloud Controller or the apps themselves.

### Enable App Traffic Logging

To enable app traffic logging:

1. From Ops Manager, navigate to the *Pivotal Application Service* tile > **Networking** pane.

2. Under *Log traffic for all accepted/denied application packets*, select **Enable (will increase log volume)** or **Disable** to enable or disable app traffic logging.

### App Logging Behavior

App traffic logging generates log messages as follows:

- **TCP traffic** - Logs the first packet of every new TCP connection.
- **UDP traffic** - Logs UDP packets sent and received, up to a maximum per-second rate for each container. Set this rate limit in the **UDP logging interval** field (default: 100).
- **Packets denied** - Logs packets blocked by either a container-specific networking policy or by Application Security Group (ASG) rules applied across the space, org, or deployment. Logs packet denials up to a maximum per-second rate for each container, set in the **Denied logging interval** field (default: 1).

### App Traffic Log Format and Contents

App traffic logs are formatted as described in the [cf-networking-release Traffic logging](https://www.pivotal.io/kb/2/133322) documentation, following the `iptables-logger` format but without line breaks. For example, the first part of an app traffic log line looks like:
Each log message includes the following:

- **Timestamp**
- **The GUID for the source or destination app that sent or was designated to receive the packet**
- **The protocol of the communication, TCP or UDP**
- **GUIDs for the container, space, and org running the source or destination app**
- **IP addresses and ports for both source and destination apps**
- A **message** field recording whether the packet was allowed or denied, with one of the following four possibilities:
  - ASG allowed packet to exit source app container
  - C2C policy allowed packet to enter destination app container
  - ASG prevented packet from exiting source app container
  - C2C policy prevented packet from entering destination app container
- **Additional information described in the cf-networking-release**

**Denied Packet Causes**

You can determine whether a denied-packet log resulted from a container networking policy or an ASG rule as follows:

- **Container networking policy:** Log message string includes ingress-denied and packet direction is ingress.
- **ASG rule:** Log message string includes egress-denied and packet direction is egress.

**Global vs. ASG-Level App Traffic Logging**

PCF supports two mechanisms for enabling app traffic logging. Setting Log traffic to Enable in Ops Manager enables app traffic logging globally for all ASGs and container policies. Setting the log property of an ASG to true enables app traffic logging at the individual ASG level.

Because these two mechanisms operate independently, PCF generates duplicate logs when app traffic logging is enabled globally and an ASG’s log property is set to true. To avoid duplicate logs, Pivotal recommends setting the log property to false for all ASGs, or leaving it out entirely, when app traffic logging is enabled globally.

To focus on specific ASGs for log analysis, Pivotal recommends enabling app traffic logs globally and using a logging platform to filter traffic logs by ASG, rather than setting log at the individual ASG level.
Configuring UAA Password Policy

Page last updated:

If your Pivotal Cloud Foundry (PCF) deployment uses the internal user store for authentication, you can configure its password policy within the Pivotal Application Service (PAS) tile.

Open the Internal UAA Configuration

1. In a browser, navigate to the fully qualified domain name (FQDN) of your Ops Manager and log in.

2. Click the Pivotal Application Service tile.

3. Select Authentication and Enterprise SSO on the Settings tab.

4. Confirm that the Internal UAA option is selected.

Set Password Requirements and Entry Attempts

1. For Minimum Password Length, enter the minimum number of characters for a valid password.

2. For Minimum Uppercase Characters Required for Password, enter the minimum number of uppercase characters required for a valid password.

3. For Minimum Lowercase Characters Required for Password, enter the minimum number of lowercase characters required for a valid password.

4. For Minimum Numerical Digits Required for Password, enter the minimum number of digits required for a valid password.

5. For Minimum Special Characters Required for Password, enter the minimum number of special characters required for a valid password.
6. For **Maximum Password Entry Attempts Allowed**, enter the maximum number of failures allowed to enter a password within a five-minute timespan before the account is locked.
Managing Internal MySQL for PAS

This topic is a reference for the concepts and procedures related to internal MySQL for Pivotal Application Service (PAS).

See the following topics:

- Scaling Internal MySQL for PAS
- Running mysql-diag
- Recovering From MySQL Cluster Downtime
- Using the MySQL Proxy
- MySQL Network Communications
Scaling Internal MySQL for PAS

This topic describes scaling down your Pivotal Application Service (PAS) MySQL cluster and provides example sizing data from two environments that have significant load on their MySQL clusters.

For additional resources about scaling Internal MySQL, see the following documentation:

- Deployments Using Internal MySQL section of Scaling PAS
- Scalable Components section of High Availability in Cloud Foundry

Note: The procedure does not apply to databases configured as external in the PAS tile Databases pane.

PAS components that use system databases include the Cloud Controller, Diego brain, Gorouter, and the User Authorization and Authentication (UAA) server. See Cloud Foundry Components.

Scale Down Your MySQL Cluster

This procedure explains how to safely scale your MySQL cluster down to a single node. If you are already running the MySQL cluster with a single node, you do not need to perform these steps.

By default, internal MySQL deploys as a single node. To take advantage of the high availability features of MySQL, you may have scaled the configuration up to three or more server nodes.

Check the Health of Your Cluster

Before scaling down your MySQL cluster, perform the following actions to ensure the cluster is healthy.

1. Use the Cloud Foundry Command Line Interface (cf CLI) to target the API endpoint of your Pivotal Cloud Foundry (PCF) deployment:

   ```bash
   $ cf api api.YOUR-SYSTEM-DOMAIN
   Setting api endpoint to api.YOUR-SYSTEM-DOMAIN... OK
   API endpoint: https://api.YOUR-SYSTEM-DOMAIN... (API version: 2.54.0)
   Not logged in. Use `cf login` to log in.
   ```

2. Log in with your User Account and Authentication (UAA) Administrator user credentials. Obtain these credentials by clicking the Credentials tab of the Pivotal Application Service (PAS) tile, locating the Admin Credentials entry in the UAA section, and clicking Link to Credential.

   ```bash
   $ cf login -u admin
   API endpoint: https://api.YOUR-SYSTEM-DOMAIN
   Password>
   Authenticating... OK
   ```

3. Create a test organization to verify the database across all nodes:

   ```bash
   $ cf create-org data-integrity-test-organization
   Creating org data-integrity-test-organization as admin... OK
   Assigning role OrgManager to user admin in org data-integrity-test-organization ... OK
   TIP: Use `cf target -o data-integrity-test-organization` to target new org
   ```

4. Obtain the IP addresses of your MySQL server by performing the following steps:
   a. From the PCF Installation Dashboard, click the Pivotal Application Service tile.
   b. Click the Status tab.
c. Record the IP addresses for all instances of the MySQL Server job.

5. Retrieve Cloud Controller database credentials from CredHub using the Ops Manager API:
   a. Perform the procedures in the Using the Ops Manager API topic to authenticate and access the Ops Manager API.
   b. Use the `GET /api/v0/deployed/products` endpoint to retrieve a list of deployed products, replacing `UAA-ACCESS-TOKEN` with the access token recorded in the previous step:

   ```
   ```
   c. In the response to the above request, locate the product with an `installation_name` starting with `cf-` and copy its `guid`.
   d. Run the following `curl` command, replacing `PRODUCT-GUID` with the value of `guid` from the previous step:

   ```
   ```
   e. Record the Cloud Controller database `username` and `password` from the response to the above request.

6. SSH into the Ops Manager VM. Because the procedures vary by IaaS, review the SSH into Ops Manager section of the Advanced Troubleshooting with the BOSH CLI topic for specific instructions.

7. For each of the MySQL server IP addresses recorded above, perform the following steps from the Ops Manager VM:
   a. Query the new organization with the following command, replacing `YOUR-IP` with the IP address of the MySQL server and `YOUR-IDENTITY` with the `identity` value of the CCDB credentials obtained above:

   ```
   $ mysql -h YOUR-IP -u YOUR-IDENTITY -D ccdb -p -e "select created_at, name from organizations where name = 'data-integrity-test-organization'"
   ```
   b. When prompted, provide the `password` value of the CCDB credentials obtained above.
   c. Examine the output of the `mysql` command and verify the `created_at` date is recent.

8. If each MySQL server instance does not return the same `created_at` result, contact Pivotal Support before proceeding further or making any changes to your deployment. If each MySQL server instance does return the same result, then you can safely proceed to scaling down your cluster to a single node by performing the steps in the following section.

Set Server Instance Count to 1

1. From the PCF Installation Dashboard, click the Pivotal Application Service tile.
2. Select Resource Config.
3. Use the dropdown to change the Instances count for MySQL Server to 1.
4. Click Save, then Review Pending Changes and Apply Changes to apply the changes.
5. Delete your test organization with the following cf CLI command:

   ```
   $ cf delete-org data-integrity-test-organization
   ```

MySQL Cluster Sizing Examples

This topic describes two sizing examples for internal MySQL in PAS.

Use this data as guidance to ensure your MySQL Clusters are scaled to handle the number of app instances running on your deployment.
Example 1: Pivotal Web Services Production Environment

The information in this section comes from the Pivotal-managed Cloud Foundry deployment, Pivotal Web Services (PWS).

Note: This deployment differs from most PCF deployments in that the MySQL database is used for Diego, but not the Cloud Controller. This means that while there are a large number of queries per second, most of them are reading data, and the number of writes is not realistically reflected.

- IAAS: AWS
- App Instances: ~23,000
- Average SQL Queries Per Minute: ~308,000
- Average IOPS: 220
- Storage Volume Usage: 80%

VM Sizing

The following table displays MySQL VM settings for this environment.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Type</td>
<td>c4.2xlarge</td>
</tr>
<tr>
<td>Storage Volume Type</td>
<td>io1</td>
</tr>
<tr>
<td>Storage Volume Size</td>
<td>98GB</td>
</tr>
<tr>
<td>Storage Volume IOPS</td>
<td>2000</td>
</tr>
</tbody>
</table>

Example 2: Cloud Foundry Diego Test Environment

The information in this section comes from an environment used by the Diego to test the MySQL cluster with a high load similar to one generated large deployment.

- IAAS: GCP
- App Instances: 250,000
- Average SQL Queries Per Minute: ~5,100,000
- Average IOPS:
  - Reads: 3.63
  - Writes: 363.98

VM Sizing

The following table displays MySQL VM settings for this environment.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Type</td>
<td>n1-standard-16</td>
</tr>
<tr>
<td>Storage Volume Type</td>
<td>pd-ssd</td>
</tr>
<tr>
<td>Storage Volume Size</td>
<td>1TB</td>
</tr>
</tbody>
</table>
Running mysql-diag

This topic discusses how to use the mysql-diag tool in MySQL for Pivotal Cloud Foundry (PCF). mysql-diag prints the state of your MySQL highly available (HA) cluster and suggests solutions if your node fails. Pivotal recommends running this tool against your HA cluster before all deployments.

mysql-diag checks the following information about the status of your HA cluster:

- Membership status of all nodes
- Size as it appears to all nodes
- If it needs to be bootstrapped
- If replication is working
- Used disk space and inodes per server

Run mysql-diag Using the BOSH Command Line Interface (CLI)

To use the BOSH CLI to run mysql-diag, do the following:

1. Obtain the information needed to use the BOSH CLI by doing the procedure in Gather Credential and IP Address Information.

2. SSH into your Ops Manager VM by doing the procedure in Log in to the Ops Manager VM with SSH for your IaaS.

3. Log in to your BOSH Director by doing the procedure in Log in to the BOSH Director.

4. Identify the VM to log in to with SSH by running the following command:

   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT vms
   ```

   Where:

   - **MY-ENV** is the name of your environment.
   - **MY-DEPLOYMENT** is the name of your deployment.

5. Record the GUID associated with the mysql-monitor VM, also known as the jumpbox VM.

6. SSH into your mysql-monitor VM by running the following command:

   ```
   bosh -e MY-ENV -d MY-DEP ssh mysql-monitor/GUID
   ```

   Where:

   - **MY-ENV** is the name of your environment.
   - **MY-DEPLOYMENT** is the name of your deployment.
   - **GUID** is the GUID you recorded in the previous step.

7. View the status of your HA cluster by running the following command:

   ```
   mysql-diag
   ```

Example Healthy Output

The mysql-diag command returns the following message if your canary status is healthy:

```
Checking canary status...healthy
```

Here is a sample mysql-diag output after the tool identified a healthy HA cluster:

---

© Copyright Pivotal Software Inc, 2013-2019
Example Unhealthy Output

The `mysql-diag` command returns the following message if your canary status is unhealthy:

```
Checking canary status...unhealthy
```

In the event of a broken HA cluster, running `mysql-diag` outputs actionable steps meant to expedite the recovery of that HA cluster. Below is a sample `mysql-diag` output after the tool identified an unhealthy HA cluster:
You must bootstrap the cluster. Follow these instructions: http://docs.pivotol.io/p/mysql/bootstrap.html

Run the download-logs command:
$ download-logs -d /tmp/output -n 10.244.7.2 -n 10.244.8.2 -n 10.244.9.2

For full information about how to download and use the download-logs command see https://discuss.pivotol.io/en-us/articles/22154488

Do not perform the following unless instructed by Pivotal Support:
- Do not scale down the cluster to one node then scale back. This puts user data at risk.
- Avoid "bash recreate" and "bash ck". These options remove logs on the WM making it harder to diagnose cluster issues.
Recovering From MySQL Cluster Downtime

Page last updated:

This topic describes how to bootstrap your MySQL cluster in the event of a cluster failure.

You can bootstrap your cluster by using one of two methods:

- Run the bootstrap errand. See Run the Bootstrap Errand below.
- Bootstrap manually. See Bootstrap Manually below.

When to Bootstrap

You must bootstrap a cluster that loses quorum. A cluster loses quorum when less than half of the nodes can communicate with each other for longer than the configured grace period. If a cluster does not lose quorum, individual unhealthy nodes automatically rejoin the cluster after resolving the error, restarting the node, or restoring connectivity.

To check whether your cluster has lost quorum, look for the following symptoms:

- All nodes appear "Unhealthy" on the proxy dashboard, viewable at https://BOSH-JOB-INDEX-proxy-p-mysql-ert.YOUR-SYSTEM-DOMAIN:

  ![Switchboard](image)

<table>
<thead>
<tr>
<th>NODES</th>
<th>STATUS</th>
<th>CURRENT SESSIONS</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>backend-0</td>
<td>UNHEALTHY</td>
<td>0</td>
<td>10.85.3.140</td>
</tr>
<tr>
<td>backend-1</td>
<td>UNHEALTHY</td>
<td>0</td>
<td>10.85.3.141</td>
</tr>
<tr>
<td>backend-2</td>
<td>UNHEALTHY</td>
<td>0</td>
<td>10.85.3.142</td>
</tr>
</tbody>
</table>

- All responsive nodes report the value of `wsrep_cluster_status` as non-Primary:

  ```
  mysql> SHOW STATUS LIKE 'wsrep_cluster_status';
  +----------------------+-------------+
  | Variable_name       | Value       |
  +----------------------+-------------+
  | wsrep_cluster_status | non-Primary |
  +----------------------+-------------+
  ```

- All unresponsive nodes respond with ERROR 1047 when using most statement types in the MySQL client:

  ```
  mysql> select * from mysql.user;
  ERROR 1047 (25000) at line 1: WSREP has not yet prepared node for application use
  ```

  For more information about checking the state of your cluster, see Check Cluster State.

Run the Bootstrap Errand

MySQL for PCF includes a BOSH errand that automates the manual bootstrapping procedure in the Bootstrap Manually section below.

It finds the node with the highest transaction sequence number and asks it to start up by itself (i.e. in bootstrap mode) and then asks the remaining nodes to join the cluster.

In most cases, running the errand recovers your cluster. But, certain scenarios require additional steps.
Determine Type of Cluster Failure

To determine which set of instructions to follow, do the following:

1. Run the following command:

   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
   ```

   Where:
   - **YOUR-ENV** is the environment where you deployed the cluster.
   - **YOUR-DEPLOYMENT** is the deployment cluster name.

   For example:

   ```bash
   $ bosh -e prod -d mysql instances
   ```

2. Find and record the Process State for your MySQL instances. In the following example output, the MySQL instances are in the **failing** process state.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Process State</th>
<th>AZ</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup-restore/a5f74106-9172-46a5-9054-8d2262c3c0</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.47</td>
</tr>
<tr>
<td>broker-register/a95e33af-c9c3-4308-8545-3b2b60594d4</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.58</td>
</tr>
<tr>
<td>cf-mysql-broker/1375225b-0a-0b-01-1f-3d-4e7f-444510f277f8</td>
<td>running</td>
<td>us-central1-c</td>
<td>10.0.4.57</td>
</tr>
<tr>
<td>cf-mysql-broker/208c6439-fc-6e-92-3c-fb-48-82a7d44f-57</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.56</td>
</tr>
<tr>
<td>deregister-and-purge-instances/a5f74106-9172-46a5-9054-8d2262c3c0</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.48</td>
</tr>
<tr>
<td>mysql/220fe72a-9026-4e2e-9fe3-1f5c0b6bf09b</td>
<td>failing</td>
<td>us-central1-b</td>
<td>10.0.4.44</td>
</tr>
<tr>
<td>mysql/28a210ac-cb98-4ab4-9672-96c61578d5</td>
<td>failing</td>
<td>us-central1-c</td>
<td>10.0.4.46</td>
</tr>
<tr>
<td>mysql/1375225b-0a-0b-01-1f-3d-4e7f-444510f277f8</td>
<td>failing</td>
<td>us-central1-b</td>
<td>10.0.4.45</td>
</tr>
<tr>
<td>mysql/28b463b1-cc12-42bf-b34b-82ca7c417c41</td>
<td>running</td>
<td>us-central1-c</td>
<td>10.0.4.40</td>
</tr>
<tr>
<td>mysql/c1639373-26a2-44ce-85d5-96f5ae290d4b</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.40</td>
</tr>
<tr>
<td>proxy/87c5683d-1253-4e26-b925-625215290dab</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.40</td>
</tr>
<tr>
<td>proxy/60115ced-797c-42e3-38e5-c63e4c50915</td>
<td>running</td>
<td>us-central1-c</td>
<td>10.0.4.40</td>
</tr>
<tr>
<td>rejoints-vm/1a493268-8a8b-4e3c-8f7e-6168350413</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.35</td>
</tr>
<tr>
<td>smoke-tests/9aa6af-ef0d-4871-9f14-081f38be773</td>
<td>running</td>
<td>us-central1-b</td>
<td>10.0.4.39</td>
</tr>
</tbody>
</table>

3. Choose your scenario:
   - If your MySQL instances are in the **failing** state, continue to **Scenario 1**.
   - If your MySQL instances are in the **-** state, continue to **Scenario 2**.

**Scenario 1: VMs Running, Cluster Disrupted**

In this scenario, the VMs are running, but the cluster has been disrupted.

To bootstrap in this scenario, follow these steps:

1. To run the bootstrap errand, run the following command:

   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT run-errand bootstrap
   ```

   Where:
   - **YOUR-ENV** is the name of your environment.
   - **YOUR-DEPLOYMENT** is the name of your deployment.

   **Note:** The errand runs for a long time, during which no output is returned.

The command returns many lines of output, eventually followed by:

```
Bootstrap errand completed
[stderr]
+ echo 'Started bootstrap errand ...'
+ JOB_DIR=/var/vcap/jobs/bootstrap/config/config.yml
+ var=vcap/packages/bootstrap/bin/cf/mysql-bootstrap -configPath=/var/vcap/jobs/bootstrap/config/config.yml
+ echo 'Bootstrap errand completed'
+ exit 0
Errand 'bootstrap' completed successfully (exit code 0)
```
2. If the errand fails, run the bootstrap errand command again after a few minutes. The bootstrap errand might not work immediately.

3. If the errand fails after several tries, bootstrap your cluster manually. See Bootstrap Manually below.

Scenario 2: VMs Terminated or Lost

In severe circumstances, such as a power failure, it is possible to lose all your VMs. You must recreate them before you can begin recovering the cluster.

When MySQL instances are in the **-state**, the VMs are lost. The procedures in this scenario bring the instances from a **-state** to a **failing** state. Then you run the bootstrap errand similar to Scenario 1 above and restore configuration.

To recover terminated or lost VMs, do the procedures in the sections below:

1. **Recreate the Missing VMs**: Bring MySQL instances from a **-state** to a **failing** state.
2. **Run the Bootstrap Errand**: Since your instances are now in the **failing** state, you continue similarly to Scenario 1 above.
3. **Restore the BOSH Configuration**: Go back to unignoring all instances and redeploy. This is a critical and mandatory step.

**warning:** If you do not set each of your ignored instances to **unignore**, your instances are not updated in future deploys. You must perform the procedure in the final section of Scenario 2, Restore the BOSH Configuration.

Recreate the Missing VMs

The procedure in this section uses BOSH to recreate the VMs, install software on them, and try to start the jobs.

The procedure below allows you to do the following:

- Redeploy your cluster while expecting the jobs to fail.
- Instruct BOSH to ignore the state of each instance in your cluster. This allows BOSH to deploy the software to each instance even if the instance is failing.

To recreate your missing VMs, do the following:

1. To SSH into the BOSH VMs, follow the procedure in BOSH SSH.
2. If BOSH resurrection is enabled, disable it by running the following command:
   ```bash
   bosh -e YOUR-ENV update-resurrection off
   ```
   Where `YOUR-ENV` is the name of your environment.
3. To download the current manifest, run the following command:
   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT manifest > /tmp/manifest.yml
   ```
   Where:
   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.
4. To redeploy deployment, run the following command:
   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT deploy /tmp/manifest.yml
   ```
   Where:
   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

**Note:** Expect one of the MySQL VMs to fail. Deploying causes BOSH to create new VMs and install the software. Forming a cluster is in a subsequent step.
5. To view the instance GUID of the VM that attempted to start, run the following command:

```
bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
```

Where:
- `YOUR-ENV` is the name of your environment.
- `YOUR-DEPLOYMENT` is the name of your deployment.

Record the instance GUID, which is the string after `mysql/` in your BOSH instances output.

6. To instruct BOSH to ignore each MySQL VM, run the following command:

```
bosh -e YOUR-ENV -d YOUR-DEPLOYMENT ignore mysql/INSTANCE-GUID
```

Where:
- `YOUR-ENV` is the name of your environment.
- `YOUR-DEPLOYMENT` is the name of your deployment.
- `INSTANCE-GUID` is the GUID of your instance you recorded in the above step.

7. Repeat steps 4 through 6 until all instances have attempted to start.

8. If you disabled BOSH resurrection in step 2, to re-enable it, run the following command:

```
bosh -e YOUR-ENV update-resurrection on
```

Where: `YOUR-ENV` is the name of your environment.

9. To confirm that your MySQL instances have gone from the `:` state to the `failing` state, run the following command:

```
bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
```

Where:
- `YOUR-ENV` is the name of your environment.
- `YOUR-DEPLOYMENT` is the name of your deployment.

Run the Bootstrap Errand

After you recreate the VMs, all instances now have a `failing` process state and have the MySQL code. You must run the bootstrap errand to recover the cluster.

To bootstrap, do the following:

1. To run the bootstrap errand, run the following command:

```
bosh -e YOUR-ENV -d YOUR-DEPLOYMENT run-errand bootstrap
```

Where:
- `YOUR-ENV` is the name of your environment.
- `YOUR-DEPLOYMENT` is the name of your deployment.

**Note:** The errand runs for a long time, during which no output is returned.

The command returns many lines of output, eventually with the following successful output:
2. If the errand fails, run the bootstrap errand command again after a few minutes. The bootstrap errand might not work immediately.

3. See that the errand completes successfully in the shell output and continue to Restore the BOSH Configuration below.

Note: After you complete the bootstrap errand, you might still see instances in the failing state. Continue to the next section anyway.

Restore the BOSH Configuration

⚠️ warning: If you do not set each of your ignored instances to unignore, your instances are never updated in future deploys.

To restore your BOSH configuration to its previous state, this procedure unignores each instance that was previously ignored:

1. For each ignored instance, run the following command:

   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT unignore mysql/INSTANCE-GUID
   ```

   Where:
   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.
   - `INSTANCE-GUID` is the GUID of your instance.

2. To redeploy your deployment, run the following command:

   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT deploy
   ```

   Where:
   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

3. To validate that all mysql instances are in a running state, run the following command:

   ```bash
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
   ```

   Where:
   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

Bootstrap Manually

If the bootstrap errand is not able to automatically recover the cluster, you might need to do the steps manually.

Do the procedures in the sections below to manually bootstrap your cluster.

⚠️ warning: The following procedures are prone to user-error and can result in lost data if followed incorrectly. Follow the procedure in Bootstrap with the BOSH Errand above first, and only resort to the manual process if the errand fails to repair the cluster.
Shut Down MySQL

Do the following for each node in the cluster:

1. To SSH into the node, follow the procedure in BOSH SSH.

2. To shut down the mysqld process on the node, run the following command:

   ```
   monit stop galera-init
   ```

Re-bootstrapping the cluster is not successful unless you shut down the mysqld process on all nodes in the cluster.

Choose Node to Bootstrap

To choose the node to bootstrap, you must find the node with the highest transaction sequence number (seqno).

Do the following to find the node with the highest seqno:

1. Run the following command.

   ```
   cat /var/vcap/store/pxc-mysql/grastate.dat | grep 'seqno:'
   ```

2. If a node shut down gracefully, the seqno is in the Galera state file. Retrieve the seqno and continue to Bootstrap the First Node.

   a. If a node crashed or was killed, the seqno in the Galera state file is recorded as -1. In this case, the seqno might be recoverable from the database. Run the following command to start up the database, log the recovered seqno, and then exit:

      ```
      /var/vcap/packages/pxc/bin/mysqld --defaults-file=/var/vcap/jobs/pxc-mysql/config/my.cnf --wsrep-recover
      ```

   b. Scan the error log for the recovered seqno. It is the last number after the group id (uuid). For example:

      ```
      $ grep "Recovered position" /var/vcap/sys/log/pxc-mysql/mysql.err.log | tail -1
      150225 18:09:42 mysqld_safe WSREP: Recovered position e93955c7-b797-11e4-9faa-9a6f0b73eb46:15
      ```

   c. If the node never connected to the cluster before crashing, it may not even have a group ID (uuid in grastate.dat). In this case, there is nothing to recover. Unless all nodes crashed this way, do not choose this node for bootstrapping.

3. After determining the seqno for all nodes in your cluster, identify the node with the highest seqno. If all nodes have the same seqno, you can choose any node as the new bootstrap node.

Bootstrap the First Node

After determining the node with the highest sequence_number, do the following to bootstrap the node:

1. To update the state file and restart the mysqld process on the new bootstrap node, run the following commands:

   ```
   echo -n "NEEDS_BOOTSTRAP" > /var/vcap/store/pxc-mysql/state.txt
   monit start galera-init
   ```

2. It can take up to ten minutes for monit to start the mysqld process. To check if the mysqld process has started successfully, run the following command:

   ```
   watch monit summary
   ```

Restart Remaining Nodes

After the bootstrapped node is running, to restart the nodes do the following:

1. To start the mysqld process with monit, run the following command:
If the node is prevented from starting by the `monit start galera-init`, do the manual procedure to force the node to rejoin the cluster, documented in Pivotal Knowledge Base.

**warning:** Forcing a node to rejoin the cluster is a destructive procedure. Only do the procedure with the assistance of Pivotal Support.

2. If the `monit start` command fails, it might be because the node with the highest `sequence_number` is `mysql/0`.

In this case, do the following:

a. To make BOSH ignore updating `mysql/0`, run the following command:

```bash
bosh -e YOUR-ENV -d YOUR-DEPLOYMENT ignore mysql/0
```

Where:

- `YOUR-ENV` is the name of your environment.
- `YOUR-DEPLOYMENT` is the name of your deployment.

b. Navigate to Ops Manager in a browser, log in, and click **Apply Changes**.

c. When the deploy finishes, run the following command from the Ops Manager VM:

```bash
bosh -e YOUR-ENV -d YOUR-DEPLOYMENT unignore mysql/0
```

Where:

- `YOUR-ENV` is the name of your environment.
- `YOUR-DEPLOYMENT` is the name of your deployment.

3. To verify that the new nodes have successfully joined the cluster, SSH into the bootstrap node and run the following command to output the total number of nodes in the cluster:

```sql
mysql> SHOW STATUS LIKE 'wsrep_cluster_size';
```
Configuring Authentication and Enterprise SSO for PAS

Page last updated:

This topic describes Pivotal Cloud Foundry (PCF) Pivotal Application Service (PAS) authentication and single sign-on configuration with Lightweight Directory Access Protocol (LDAP) and Security Assertion Markup Language (SAML).

Refer to the instructions below to configure your deployment with SAML or LDAP.

Connecting Pivotal Application Service (PAS) to either the LDAP or SAML external user store allows the User Account and Authentication (UAA) server to delegate authentication to existing enterprise user stores.

If your enterprise user store is exposed as a SAML or LDAP Identity Provider for single sign-on (SSO), you can configure SSO to allow users to access the Apps Manager and Cloud Foundry Command Line Interface (cf CLI) without creating a new account or, if using SAML, without re-entering credentials.

See the Adding Existing SAML or LDAP Users to a PCF Deployment topic for information about managing user identity and pre-provisioning user roles with SAML or LDAP in PCF.

For an explanation of the process used by the UAA Server when it attempts to authenticate a user through LDAP, see Configuring LDAP Integration with Pivotal Cloud Foundry in the Pivotal Knowledge Base.

Note: When integrating with an external identity provider, such as LDAP, authentication within the UAA becomes chained. An authentication attempt with a user's credentials is first attempted against the UAA user store before the external provider, LDAP. For more information, see Chained Authentication in the User Account and Authentication LDAP Integration GitHub documentation.

Configure PAS to Use a SAML Identity Provider

In SAML terminology, the SAML protocol communicates user data between an identity provider (IdP) and a service provider (SP).

To connect PAS with SAML, do the following:

- Configure PAS as a Service Provider for SAML
- Configure SAML as an Identity Provider for PAS

Configure PAS as a Service Provider for SAML

To configure PAS to use a SAML IdP, do the following:

1. From the Installation Dashboard, click the PAS tile.
2. Select the Domains tab and record your system domain.
3. Select Authentication and Enterprise SSO.
4. Select SAML Identity Provider.

5. Set the Provider Name. This is a unique name you create for the Identity Provider. This name can include only alphanumeric characters, +, _, and -. You should not change this name after deployment because all external users use it to link to the provider.

6. Enter a Display Name. Your provider display name appears as a link on your Pivotal login page, which you can access at https://login.YOUR-SYSTEM.
7. Retrieve the metadata from your Identity Provider and copy it into either the Provider Metadata or the Provider Metadata URL fields, depending on whether your Identity Provider exposes a Metadata URL or not. Refer to the Configure SAML as an Identity Provider for PAS section of this topic for more information. Pivotal recommends that you use the Provider Metadata URL rather than Provider Metadata because the metadata can change. You can do this in either of the following ways:
   - If your Identity Provider exposes a Metadata URL, provide the Metadata URL.
   - Download your Identity Provider metadata and paste this XML into the Provider Metadata area.

   **Note:** You only need to select one of the above configurations. If you configure both, your Identity Provider defaults to the (OR) Provider Metadata URL.

   **Note:** See Adding Existing SAML or LDAP Users to a PCF Deployment for information about on-boarding SAML users and mapping them to PAS user roles.

8. Select the **Name ID Format** for your SAML Identity Provider. This translates to **username** on PAS. The default is **Email Address**.

9. For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who will receive invitations to Apps Manager.

10. For **First Name Attribute** and **Last Name Attribute**, enter the attribute names in your SAML database that correspond to the first and last names in each user record, for example **first_name** and **last_name**. This field is case sensitive.

11. For **Email Attribute**, enter the attribute name in your SAML assertion that corresponds to the email address in each user record, for example **EmailID**. This field is case sensitive.

12. For **External Groups Attribute**, enter the attribute name in your SAML database that defines the groups that a user belongs to, for example **group_memberships**. To map the groups from the SAML assertion to admin roles in PAS, follow the instructions in the Grant Admin Permissions to an External Group (SAML or LDAP) section of the Creating and Managing Users with the UAA CLI (UAAC) topic. This field is case sensitive.

13. By default, all SAML Authentication Request from PAS are signed. To change this, disable the **Required Signed Assertions** checkbox and configure your Identity Provider to verify SAML authentication requests.

14. To validate the signature for the incoming SAML assertions, enable the **Required Signed Assertions** checkbox and configure your Identity Provider to send signed SAML assertions.

15. Click **Save**.

16. Return to the **Installation Dashboard** by clicking the link.

17. On the Installation Dashboard, click **Review Pending Changes**, then **Apply Changes**.
Configure SAML as an Identity Provider for PAS

The following procedure configures a SAML IdP to designate PAS as an SP.

Download the Service Provider Metadata from https://login.YOUR-SYSTEM-DOMAIN/saml/metadata. Consult the documentation from your Identity Provider for configuration instructions.

Refer to the table below for information about certain industry-standard Identity Providers and how to integrate them with PAS:

<table>
<thead>
<tr>
<th>Solution Name</th>
<th>Integration Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA Single Sign-On aka CA SiteMinder</td>
<td>Link</td>
</tr>
<tr>
<td>Ping Federate</td>
<td>Link</td>
</tr>
<tr>
<td>Active Directory Federation Services</td>
<td>Link</td>
</tr>
</tbody>
</table>

Note: Some Identity Providers allow uploads of Service Provider Metadata. Other providers require you to manually enter the Service Provider Metadata into a form. If your Identity Provider requires manual entry but is not listed above, see the CA SiteMinder SSO Integration Guide.

Configure LDAP as an Identity Provider for PAS

To integrate the UAA with one or more LDAP servers, configure PAS with your LDAP endpoint information as follows:

1. Log into the Operations Manager web interface.
2. On the Product Dashboard, select the PAS tile.
3. In the left navigation menu, select Authentication and Enterprise SSO.
4. Under **Configure your UAA**, select **LDAP Server**.

- For **Server URL**, enter the URL that points your LDAP server. With multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
  - `ldap://` This specifies that the LDAP server uses an unencrypted connection.
  - `ldaps://` This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or that you import a trusted certificate to the JVM truststore.

- For **LDAP Credentials**, enter the LDAP Distinguished Name (DN) and password for binding to the LDAP Server. Example DN:
  
  Note: Pivotal recommends that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base. In addition to this, if the bind user belongs to a different search base, you must use the full DN.

  **warning:** Pivotal recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

- For **User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

  For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`
For **User Search Filter**, enter a string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

In the LDAP search filter string that you use to configure PAS, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

In addition to `cn`, other attributes commonly searched for and returned are `mail`, `uid`, and, in the case of Active Directory, `sAMAccountName`.

**Note:** For instructions for testing and troubleshooting your LDAP search filters, see the Configuring LDAP Integration with Pivotal Cloud Foundry Knowledge Base article.

For **Group Search Base**, enter the location in the LDAP directory tree from which the LDAP Group search begins.

For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base: `ou=Groups,dc=example,dc=com`.

Follow the instructions in the Grant Admin Permissions to an External Group (SAML or LDAP) section of Creating and Managing Users with the UAA CLI (UAAC) to map the groups under this search base to admin roles in PAS.

**Note:** See Adding Existing SAML or LDAP Users to a PCF Deployment to on-board individual LDAP users and map them to PAS Roles.

For **Group Search Filter**, enter a string that defines LDAP Group search criteria. The standard value is `member={0}`.

For **Server SSL Cert**, paste in the root certificate from your CA certificate or your self-signed certificate.

For **First Name Attribute** and **Last Name Attribute**, enter the attribute names in your LDAP directory that correspond to the first and last names in each user record, for example `cn` and `sn`.

For **Email Attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record, for example `mail`.

For **Email Domain(s)**, enter a comma-separated list of the email domains for external users who will receive invitations to Apps Manager.

1. For **LDAP Referrals**, select how the UAA handles LDAP server referrals out to other external user stores. The UAA can:
   - Automatically follow any referrals.
   - Ignore referrals and return partial result.
   - Throw exception for each referral and abort.

   **Click** Save.

   **Return to the** Installation Dashboard by clicking the link.

   **On the Installation Dashboard, click** Review Pending Changes, then **Apply Changes**.
Configuring ADFS as an Identity Provider

Page last updated:

This topic describes the process of configuring Active Directory Federation Services (ADFS) as your identity provider (IdP) in Pivotal Cloud Foundry (PCF) and ADFS.

Configure SAML Integration in PCF

You can use ADFS as your SAML IdP for Ops Manager and Pivotal Application Service (PAS):

- If you want to use ADFS as your SAML IdP for Ops Manager, do the following:
  - Configure SAML Integration in Ops Manager
  - Configure SAML Integration in ADFS

- If you want to use ADFS as your SAML IdP for PAS, do the following:
  - Configure SAML Integration in PAS
  - Configure SAML Integration in ADFS

Configure SAML Integration in Ops Manager

To configure Ops Manager to use ADFS as your SAML IdP, do the following:


2. Perform the steps in the Use an Identity Provider section of the BOSH Director configuration topic for your IaaS:
   - Configuring BOSH Director on AWS
   - Configuring BOSH Director on Azure Manually
   - Configuring BOSH Director on GCP
   - Configuring BOSH Director on OpenStack
   - Configuring BOSH Director on vSphere

Note: You can set up SAML access for Ops Manager during the initial PCF installation or later by navigating to Settings in the user menu on the Ops Manager Installation Dashboard, configuring the Authentication Method pane, and then clicking Review Pending Changes and Apply Changes.

Configure SAML Integration in PAS

To configure PAS to use ADFS as your SAML IdP, do the following:


2. Perform the steps in the Configure PCF as a Service Provider for SAML section of the Configuring Authentication and Enterprise SSO for PAS topic.

Configure SAML Integration in ADFS

To designate PCF as your SAML service provider (SP) in ADFS, do the following:


2. Open your ADFS Management console and add a relying party trust as follows:
   a. Click Add Relying Party Trust... in the Actions pane.
   b. On the Welcome step, click Start.
   c. Select Import data about the relying party from a file, import the downloaded SP metadata file, and click Next.
   d. Enter a Display name for the new relying party trust and click Next.
e. Leave the default multi-factor authentication selection and click Next.
f. Select Permit all users to access this relying party and click Next.
g. Review your settings and click Next.
h. Click Close to finish the wizard.

3. Modify your relying party trust as follows:

a. Double-click the new relying party trust.
b. Select the Encryption tab and click Remove to remove the encryption certificate you imported.
c. In the Advanced tab, select SHA256 for the Secure hash algorithm.

4. (Optional) If you are using a self-signed certificate and want to disable CRL checks, do the following:

a. Open Windows Powershell as an Administrator.
b. Execute the following command:

```
set-ADFSRelyingPartyTrust -TargetName "RELYING-PARTY-TRUST" -SigningCertificateRevocationCheck None
```

5. To add claim rules for your relying party trust, select your relying party trust and click Edit Claim Rules....

6. In the Issuance Transform Rules tab, create two claim rules as follows:

a. Click Add Rule.
b. Select Send LDAP Attributes as Claims for Claim rule template and click Next.
c. Enter a Claim rule name.
d. Select Active Directory for Attribute store.
e. Select E-Mail-Addresses for LDAP Attribute and E-Mail Address for Outgoing Claim Type. Alternatively, if you do not have the email attribute configured for users, you can select User-Principal-Name under LDAP Attribute.
f. Click Finish.

g. Click Add Rule.
h. Select Transform an Incoming Claim for Claim rule template and click Next.
i. Enter a Claim rule name.
j. Select E-Mail Address for Incoming claim type.
k. Select Name ID for Outgoing claim type.
l. Select Email for Outgoing name ID format.
m. Click Finish.

7. To permit access to users based on a security group, navigate to the Issuance Authorization Rules tab and create an authorization claim rule as follows:

a. Click Add Rule.
b. Select Permit or Deny Users Based on an Incoming Claim for Claim rule template and click Next.
c. Enter a Claim rule name.
d. Select Group SID for Incoming claim type.
e. Click Browse and locate the security group in your domain that PCF developers are a part of and click OK.
f. Ensure Permit access to users with this incoming claim is selected.
g. Click Finish.
Configuring CA as an Identity Provider

This topic explains how to configure single sign-on (SSO) between CA and Pivotal Cloud Foundry (PCF).

Partnership creation between CA and PCF involves the following steps:

1. Installing and configuring the prerequisites
2. Configuring CA Single Sign-On as an Identity Provider
3. Configuring the Service Provider

Prerequisites

- CA Single Sign-On v12.52 installation
- User store and session store configuration
- Creation of Signed Certificate by a Certificate Authority
- Protect Identity Provider URL with CA SSO by creating the following objects:
  - Authentication scheme
  - Domain
  - Realm
  - Rules and policy
- PCF environment at https://console.YOUR-SYSTEM-DOMAIN

Note: Replace YOUR-SYSTEM-DOMAIN with the name of your PCF installation.

Configuring CA as the SAML 2.0 Identity Provider on PCF

1. Download the service provider metadata.
   a. Navigate [https://login.YOUR-SYSTEM-DOMAIN/saml/metadata](https://login.YOUR-SYSTEM-DOMAIN/saml/metadata) and log in to CA SSO.
   b. Navigate to Federation.
   c. Select Partnership Federation
   d. In the Actions menu, select Export Metadata.
   e. Save the exported metadata in an XML file.

2. Follow the steps in Configuring Authentication and Enterprise SSO for Elastic Runtime to set the identity provider metadata on PCF.

3. Paste the contents of the XML file into the Identity Provider Metadata field.

4. Click Save.

5. Click Review Pending Changes, then Apply Changes.

Configuring PCF as the SAML 2.0 Service Provider on CA Single Sign-On

Configure Identity Provider and Service Provider Entities

1. Navigate to [https://login.YOUR-SYSTEM-DOMAIN](https://login.YOUR-SYSTEM-DOMAIN) and log in to CA SSO.
2. Navigate to Federation.
3. Click Partnership Federation.
4. Click Entity.
5. Click **Create Entity**.

6. To create a local entity, use the values below:
   - **Entity Location**: Local
   - **Entity Type**: SAML2 IDP
   - **Entity ID**: Enter an ID for your local identity provider. For example, [https://ca-technologies.xxx.com](https://ca-technologies.xxx.com).
   - **Entity Name**: Create a name for your local identity provider.
   - **Base URL**: Enter the fully-qualified domain name for the host service CA SSO Federation Web Services.
   - **Signing Private Key Alias**: Select the private key alias or import a private key.
   - **Signed Authentication Requests Required**: Select **No**.
   - **Supported NameID format**: Enter `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress` and `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` to select both email address and unspecified as supported NameID formats.

7. To create a remote entity, click **Import Metadata Button** and do the following:
   a. Download the service provider metadata from [https://login.{systemdomain}/saml/metadata](https://login.{systemdomain}/saml/metadata) and save to an XML file.
   b. Browse and select the saved XML Metadata you downloaded in the previous step.
   c. Provide a name for the Remote Service Provider Entity.
   d. Provide an alias for the Signing Certificate imported from the metadata.
   e. Click **Save**.

   **Note:** PCF signs the outgoing SAML authentication requests.

**Configure Partnership Between CA SSO and PCF**

1. Navigate to [https://login.YOUR-SYSTEM-DOMAIN](https://login.YOUR-SYSTEM-DOMAIN) and log in to CA SSO.

2. Navigate to **Federation**.

3. Click **Partnership Federation**.

4. Click **Create Partnership**.

5. To configure the partnership, use the values below to fill in the fields:
   - **Add Partnership Name**: Enter a name for your partnership.
   - **(optional) Description**: Enter a relevant description for your partnership.
   - **Local IPD ID**: Enter the Local Service Provider ID you created in the Configure Identity Provider and Service Provider Entities section.
   - **Remote SP ID**: Enter the Remote SP ID you created in the Configure Identity Provider and Service Provider Entities section.
   - **Base URL**: This field will be pre-populated.
   - **Skew Time**: Enter any skew time required by your environment.
   - **User Directories and Search Order**: Select the required directories in the required search order.

6. Click **Next**.

7. On the Federation Users page, accept the default values.

8. Click **Next**.

9. To complete the Name ID Format section:
   a. Select **Email Address** from the Name ID Format dropdown.
   b. Select **User Attribute** from the Name ID Type dropdown.

   **Note:** PCF does not support processing SAML Assertion Attributes at this time. You can skip filling out the Assertion Attributes fields.

10. Click **Next**.

11. To complete the SSO and SLO section:
   a. Enter the Authentication URL that is protected by CA SSO under prerequisites.
   b. For SSO Binding, click **HTTP-POST**.
   c. In the Audience field, enter [http://login.YOUR-SYSTEM-DOMAIN](http://login.YOUR-SYSTEM-DOMAIN).

© Copyright Pivotal Software Inc, 2013-2019
d. Select Both IDP and DP Initiated from the Transactions Allowed dropdown.
e. The Assertion Consumer Service URL field will be pre-populated using information from the service provider entity.

12. Click Next.

13. To complete the Configure Signature and Encryption section:
   a. In the Signing Private Key Alias dropdown, verify that the correct Private Key Alias is selected.
   b. Verify that the correct Verification Certificate Alias is selected in the Verification Certificate Analysis dropdown. This alias should be the same certificate created when you import the Remote Service Provider Entity ID.
   c. Select Sign Both from the Post Signature Options dropdown.

   Note: PCF does not support encryption options at this time.

d. Click Finish.

14. To activate the partnership, expand the Action dropdown for your partnership and click Activate.
Configuring PingFederate as an Identity Provider

This topic explains how to configure single sign-on (SSO) between PingFederate and Pivotal Application Service (PAS).

Configuring PingFederate as the SAML 2.0 Identity Provider on PAS

1. Download your Identity Provider Metadata from PingFederate Server. Click Metadata Export under Administrative Functions on the Main Menu of the PingFederate Administrative Console. If your PingFederate server is configured to act as both an Identity Provider (IdP) and a service provider (SP), indicate which type of configuration you want to export and click Next. The Signing key can be exported. You can skip the options related to Encryption Keys and Metadata Attribute Contract because they are not supported at this time.

2. Follow the steps in Configuring Authentication and Enterprise SSO for PAS to set your IdP metadata on PAS.

Configuring PAS as the SAML 2.0 Service Provider on PingFederate

1. Download your Service Provider Metadata from https://login.YOUR-SYSTEM-DOMAIN/saml/metadata .

2. Import the Service Provider Metadata to PingFederate. Navigate to Main Menu → IdP Configuration → SP Connection and click Import. In the Import Connection screen, browse and select the .xml file downloaded in the previous step. Click Import and Done.

3. PAS expects the NameID format to be an email address (for example, urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ) and the value to be the email address of the currently logged in user. The SSO does not function without this setting.
   a. Click the connection name on Main Menu. To see a full list of connections, click Manage All SP.
   b. Click Browser SSO under the SP Connection tab.
   c. Click Configure Browser SSO.
   d. Click Assertion Creation under the Browser SSO tab.
   e. Click Configure Assertion Creation.
   f. Click Identity Mapping on the Summary screen.
   g. Select Standard as the option and map the NameID format to be an email address and the value to be the email address of the user.

4. Select the Authentication Source.
   a. Click Browser SSO under the SP Connection tab.
   b. Click Configure Browser SSO.
   c. Click Assertion Creation under the Browser SSO tab.
   d. Click Configure Assertion Creation.
   e. Click IdP Adapter Mapping on the Summary screen.
   f. Click Adapter Instance Name.
   g. Click Adapter Instance on the Summary screen.

5. Enable the SSO Browser Profiles.
   a. Click Browser SSO under the SP Connection tab.
   b. Click Configure Browser SSO.
   c. Click SAML Profiles on the Summary screen.
   d. Ensure that IdP & SP initiated SSO are selected.

   Note: PAS does not support SLO profiles at this time, and you can leave them unchecked.

6. Activate the SP Connection.
Switching Application Domains

This topic describes how to change the domain of an existing Pivotal Cloud Foundry (PCF) installation, using an example domain change from myapps.mydomain.com to newapps.mydomain.com.

1. In PCF Ops Manager, select the Pivotal Application Service tile.

2. Select Domains from the menu to see the current Apps Domain for your Pivotal Application Service (PAS) deployment. In the following example it is myapps.mydomain.com.

3. In the terminal, run `cf login -a YOUR_API_ENDPOINT`. The cf CLI prompts you for your PCF username and password, as well as the org and space you want to access. See Identifying the API Endpoint for your PAS Instance if you don’t know your API endpoint.

4. Run `cf domains` to view the domains in the space. If you have more than one shared domain, ensure that the domain you want to change is at the top of the list before you apply the new domain to your PAS tile configuration. You can delete and re-create the other shared domains as necessary to push the domain you want to change to the top of the list. If you do this, make sure to re-map the routes for each domain.

```
cf domains
Getting domains in org my-org as admin...

name     status
myapps.mydomain.com  shared
```

5. Run `cf routes` to confirm that your apps are assigned to the domain you plan to change.

```
cf routes
Getting routes as admin ...

space   host    domain    apps
my-space myapp myapps.mydomain.com myapp
```

6. Run `cf create-shared-domain YOUR_DESIRED_NEW_DOMAIN` to create the new domain you want to use:

```
cf create-shared-domain newapps.mydomain.com
Creating shared domain newapps.mydomain.com as admin... OK
```

7. Run `cf map-route APP_NAME NEW_DOMAIN -a HOST_NAME` to map the new domain to your app. In this example both the NEW_DOMAIN and HOST_NAME arguments are `myapp`, since this is both the name of the app to which we are mapping a route, and the intended hostname for the URL.

```
cf map-route myapp newapps.mydomain.com -a myapp
```
8. Repeat the previous step for each app in this space. Afterwards, check Apps Manager to confirm that the route URL has updated correctly for each app:

```
8
```

9. Repeat the above steps for each space in your PCF installation except for the System org, beginning with logging into the org and space and ending with confirming the URL update.

```
9
```

10. Once you have confirmed that every app in every space has been mapped to the new domain, delete the old domain by running `cf delete-shared-domain OLD_DOMAIN_TO_DELETE`:

```
10
```

11. Configure your PAS tile to use the new domain, and apply changes. Apps that you push after your update finishes use this new domain.

```
11
```
Scaling PAS

Page last updated:

This topic discusses how to scale Pivotal Application Service (PAS) for different deployment scenarios. To increase the capacity and availability of the Pivotal Cloud Foundry (PCF) platform, and to decrease the chances of downtime, you can scale a deployment up using the instructions below.

If you want to make a PCF configuration highly available, see the High Availability in Cloud Foundry topic.

Note: In PCF v1.11 and later, PAS defaults to a highly available resource configuration.

Scaling Recommendations

The following table provides recommended instance counts for a high-availability deployment and the minimum instances for a functional deployment:

<table>
<thead>
<tr>
<th>Pivotal Application Service (PAS) Job</th>
<th>Recommended Instance Number for HA</th>
<th>Minimum Instance Number</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diego Cell</td>
<td>≥ 3</td>
<td>1</td>
<td>The optimal balance between CPU/memory sizing and instance count depends on the performance characteristics of the apps that run on Diego cells. Scaling vertically with larger Diego cells makes for larger points of failure, and more apps go down when a cell fails. On the other hand, scaling horizontally decreases the speed at which the system re-balances apps. Re-balancing 100 cells takes longer and demands more processing overhead than re-balancing 20 cells.</td>
</tr>
<tr>
<td>Diego Brain</td>
<td>≥ 2</td>
<td>1</td>
<td>For high availability, use at least one per AZ, or at least two if only one AZ.</td>
</tr>
<tr>
<td>Diego BBS</td>
<td>≥ 2</td>
<td>1</td>
<td>For high availability in a multi-AZ deployment, use at least one instance per AZ. Scale Diego BBS to at least two instances for high availability in a single-AZ deployment.</td>
</tr>
<tr>
<td>MySQL Server</td>
<td>3</td>
<td>1</td>
<td>If you use an external database in your deployment, then you can set the MySQL Server instance count to 0. For instructions about scaling down an internal MySQL cluster, see Scaling Down Your MySQL Cluster.</td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>2</td>
<td>1</td>
<td>If you use an external database in your deployment, then you can set the MySQL Proxy instance count to 0.</td>
</tr>
<tr>
<td>NATS Server</td>
<td>≥ 2</td>
<td>1</td>
<td>In a high availability deployment, you might run a single NATS instance if your deployment lacks the resources to deploy two stable NATS servers. Components using NATS are resilient to message failures and the BOSH Resurrector recovers the NATS VM quickly if it becomes non-responsive.</td>
</tr>
<tr>
<td>Cloud Controller</td>
<td>≥ 2</td>
<td>1</td>
<td>Scale the Cloud Controller to accommodate the number of requests to the API and the number of apps in the system.</td>
</tr>
<tr>
<td>Clock Global</td>
<td>≥ 2</td>
<td>1</td>
<td>For a high availability deployment, scale the Clock Global job to a value greater than 1 or to the number of AZs you have.</td>
</tr>
<tr>
<td>Router</td>
<td>≥ 2</td>
<td>1</td>
<td>Scale the router to accommodate the number of incoming requests. Additional instances increase available bandwidth. In general, this load is much less than the load on Diego cells.</td>
</tr>
<tr>
<td>UAA</td>
<td>≥ 2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Doppler Server</td>
<td>≥ 2</td>
<td>1</td>
<td>Deploying additional Doppler servers splits traffic across them. For a high availability deployment, Pivotal recommends at least two per Availability Zone.</td>
</tr>
<tr>
<td>Loggregator Traffic Controller</td>
<td>≥ 2</td>
<td>1</td>
<td>Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner. For a high availability deployment, Pivotal recommends at least two per Availability Zone.</td>
</tr>
<tr>
<td>Syslog Scheduler</td>
<td>≥ 2</td>
<td>1</td>
<td>The Syslog Scheduler is a scalable component. For high availability, use at least one instance per AZ, or at least two instances if only one AZ is present.</td>
</tr>
</tbody>
</table>

Scaling Up PAS

To scale up PAS instances, do the following:

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the PAS tile in the Installation Dashboard.


<table>
<thead>
<tr>
<th>Job</th>
<th>Instances</th>
<th>Persistent Disk Type</th>
<th>VM Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consul</td>
<td>Automatic: 3</td>
<td>Automatic: 1 GB</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>NATS</td>
<td>Automatic: 2</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>File Storage</td>
<td>Automatic: 1</td>
<td>Automatic: 100 GB</td>
<td>Automatic (cpu: 1, ram: 4 GB, disk: 64 GB)</td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>Automatic: 2</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>MySQL Server</td>
<td>Automatic: 3</td>
<td>Automatic: 100 GB</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Backup Prepare Node</td>
<td>Automatic: 1</td>
<td>Automatic: 200 GB</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Diego BBS</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>UAAR</td>
<td>Automatic: 2</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Cloud Controller</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>HAProxy</td>
<td>Automatic: 1</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Router</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>NSXv Storage</td>
<td>Automatic: 1</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Clock Global</td>
<td>Automatic: 1</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Cloud Controller/Worker</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Diego Brain</td>
<td>Automatic: 3</td>
<td>Automatic: 1 GB</td>
<td>Automatic (cpu: 1, ram: 2 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Diego Cell</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Loggregator Trafficcontroller</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>syslog-Adaptor</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>syslog-Scheduler</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>Doppler Server</td>
<td>Automatic: 3</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>TCP Router</td>
<td>Automatic: 1</td>
<td>Automatic: 1 GB</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
<tr>
<td>CreditHub</td>
<td>Automatic: 2</td>
<td>None</td>
<td>Automatic (cpu: 1, ram: 1 GB, disk: 8 GB)</td>
</tr>
</tbody>
</table>

4. To scale your deployment horizontally, increase the number of Instances of a job. See Scaling Recommendations for guidance about the number of job instances required to ensure high availability.

Note: In PCF v1.12, you cannot scale the Autoscaler job to greater than one instance.

5. To scale your deployment vertically, adjust the Persistent Disk Type and VM Type of a job to allocate more disk space and memory. If you choose Automatic from the dropdown, PAS uses the recommended amount of resources for the job.

6. Click Save.

7. Return to the Installation Dashboard, click Review Pending Changes, and click Apply Changes.

**Scaling Down PAS**

If you are deploying a PCF configuration that does not need to be highly available, Pivotal recommends scaling down job instances to the minimum number required for a functional deployment.

To scale down your deployment, do the following:

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the PAS tile in the Installation Dashboard.
4. In the Resource Config screen, decrease the number of Instances for each job. Choose the suggested values outlined in Scaling Recommendations, or in the Scaling Recommendations for Specific Deployment Configurations and click Save.

5. Return to the Installation Dashboard, click Review Pending Changes, and click Apply Changes.

Scaling Down Jobs with Persistent Disk

If you scale down or delete a job that uses persistent disk, PAS marks the disk as orphaned. Orphaned disks are not attached to any job, and PAS deletes them after five days.

Use the BOSH CLI to list and recover orphaned disks. Follow the instructions in the Prepare to Use the BOSH CLI section of the Advanced Troubleshooting with the BOSH CLI topic to log in to the BOSH Director, and then follow the procedures in Orphaned Disks in the BOSH documentation.

Scaling Recommendations for Specific Deployment Configurations

If you use one of the following configurations, choose the values in the corresponding table to scale instances for your particular deployment:

- Deployments Using External Databases
- Deployments Using Internal MySQL
- Deployments Using an External Blobstore
- Deployments Using External Load Balancers

Deployments Using External Databases

If you use an external database, you can scale down the instance counts for internal MySQL jobs.

Select the following values in the Resource Config:

<table>
<thead>
<tr>
<th>Job</th>
<th>Instance Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consul</td>
<td></td>
</tr>
<tr>
<td>NATS</td>
<td></td>
</tr>
<tr>
<td>File Storage</td>
<td></td>
</tr>
<tr>
<td>NySQL Proxy</td>
<td></td>
</tr>
<tr>
<td>NySQL Server</td>
<td></td>
</tr>
<tr>
<td>Backup Prepare Node</td>
<td></td>
</tr>
<tr>
<td>Diego BMS</td>
<td></td>
</tr>
<tr>
<td>UAA</td>
<td></td>
</tr>
<tr>
<td>Cloud Controller</td>
<td></td>
</tr>
<tr>
<td>HAProxy</td>
<td></td>
</tr>
<tr>
<td>Router</td>
<td></td>
</tr>
<tr>
<td>NySQL Monitor</td>
<td></td>
</tr>
<tr>
<td>Clock Global</td>
<td></td>
</tr>
<tr>
<td>Cloud Controller/Worker</td>
<td></td>
</tr>
<tr>
<td>Diego Brain</td>
<td></td>
</tr>
<tr>
<td>Diego Cell</td>
<td></td>
</tr>
<tr>
<td>Loggregator Trafficcontroller</td>
<td></td>
</tr>
<tr>
<td>Syslog Adapter</td>
<td></td>
</tr>
<tr>
<td>Syslog Scheduler</td>
<td></td>
</tr>
<tr>
<td>Doppler Server</td>
<td></td>
</tr>
<tr>
<td>TCP Router</td>
<td></td>
</tr>
<tr>
<td>CreditHub</td>
<td></td>
</tr>
</tbody>
</table>
Deployments Using Internal MySQL

If you use the internal MySQL database on a clean install, or on an upgrade from a configuration that previously used internal MySQL databases, you do not need to change the default values shown in the table below.

If you need to change back to this configuration, choose the values shown below in the Resource Config.

Note: Changing back to this configuration deletes any data written to your other database option.

<table>
<thead>
<tr>
<th>Job</th>
<th>Instance Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL Server</td>
<td>3</td>
</tr>
<tr>
<td>MySQL Proxy</td>
<td>2</td>
</tr>
</tbody>
</table>

Note: Apps that do not use MySQL for PCF are not affected by the scaling process when you redeploy PAS. In addition, redeploying PAS with the MySQL cluster means that the PCF API will not be available for a brief period of time. For example, you are not able to push apps or query their state during this time.

Note: For MySQL high availability, you need to configure an external load balancer. For more information about configuring a load balancer for MariaDB-based deployments, see the Configure a Load Balancer section of the Installing MySQL for PCF topic.

Deployments Using an External Blobstore

If you use an external blobstore, choose the following value in the Resource Config:

<table>
<thead>
<tr>
<th>Job</th>
<th>Instance Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Storage</td>
<td>0</td>
</tr>
</tbody>
</table>

Deployments Using External Load Balancers

If you use an external load balancer, choose the following values in the Resource Config:

<table>
<thead>
<tr>
<th>Job</th>
<th>Instance Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAProxy</td>
<td>0</td>
</tr>
<tr>
<td>Router</td>
<td>≥ 1</td>
</tr>
<tr>
<td>Diego Brain</td>
<td>≥ 1</td>
</tr>
</tbody>
</table>

For more information about configuring load balancers in the Resource Config section of PAS, see Configuring Load Balancing for PAS.
Using Docker Registries

This topic describes how to configure Pivotal Cloud Foundry (PCF) to access Docker registries such as Docker Hub, by using either a root certificate authority (CA) certificate or by adding its IP address to a whitelist. It also explains how to configure PCF to access Docker registries through a proxy.

Docker registries store Docker container images. PCF uses these images to create the Docker containers that it runs apps in.

Prerequisite: Enable Docker Support

PCF can only access Docker registries if an operator has enabled Docker support with the `cf enable-feature-flag diego_docker` command, as described in the Using Docker in Cloud Foundry topic.

With Docker enabled, developers can push an app with a Docker image using the Cloud Foundry Command Line Interface (cf CLI). For more information, see the Deploying an App with Docker topic.

Use a CA Certificate

If you provide your root CA certificate in the Ops Manager configuration, follow this procedure:

1. In the Ops Manager Installation Dashboard, click the BOSH Director tile.

2. Click Security.

3. In the Trusted Certificates field, paste one or more root CA certificates. The Docker registry does not use the CA certificate itself but uses a certificate that is signed by the CA certificate.

4. Click Save.

5. Choose one of the following:
   - If you are configuring Ops Manager for the first time, return to your specific IaaS installation instructions (AWS, Azure, GCP, OpenStack, vSphere) to continue the installation process.
If you are modifying an existing Ops Manager installation, return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes**.

After configuration, BOSH propagates your CA certificate to all application containers in your deployment. You can then push and pull images from your Docker registries.

### Use an IP Address Whitelist

If you choose not to provide a CA certificate, you must provide the IP address of your Docker registry.

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Application Service tile, and navigate to the Application Containers tab.

### Enable microservice frameworks, private Docker registries, and other services that support your applications at a container level.

- Enable Custom Buildpacks
- Allow SSH access to app containers
- Enable SSH when an app is created

#### Router application identity verification

- Router uses TLS to verify application identity (Default)
- Router and applications use mutual TLS to verify each other’s identity (incompatible with TCP Routing and SSH access to app containers)

#### Private Docker Insecure Registry Whitelist

- 10.10.10.8888.example.com:8888

#### Docker Images Disk-Cleanup Scheduling on Cell VMs

- Never clean up Cell disk-space
- Routinely clean up Cell disk-space
- Clean up disk-space once usage fills disk

#### Reserved amount of Disk (MB) (min: 1)

- 15360

3. Select **Allow SSH access to app containers** to enable app containers to accept SSH connections. If you use a load balancer instead of HAProxy, you must open port 2222 on your load balancer to enable SSH traffic. To open an SSH connection to an app, a user must have Space Developer privileges for the space where the app is deployed. Operators can grant those privileges in Apps Manager or using the cf CLI.

4. For **Private Docker Insecure Registry Whitelist**, provide the hostname or IP address and port that point to your private Docker registry. For example,
Enter 198.51.100.1:80 or mydockerregistry.com:80. Enter multiple entries in a comma-delimited sequence. SSL validation is ignored for private Docker image registries secured with self-signed certificates at these locations.

5. **Under Docker Images Disk-Cleanup Scheduling on Cell VMs**, choose one of the options listed below. For more information about these options, see [Configuring Docker Images Disk-Cleanup Scheduling](#).

   - Never clean up Cell disk-space
   - Routinely clean up Cell disk-space
   - **Clean up disk-space once threshold is reached**. If you choose this option, enter the amount of disk space limit the Cell must reach before disk cleanup initiates under Threshold of Disk-Used (MB).

6. Click **Save**.

7. Choose one of the following:

   - If you are configuring Pivotal Application Service (PAS) for the first time, return to your specific IaaS installation instructions ([AWS](#), [Azure](#), [GCP](#), [OpenStack](#), [vSphere](#)) to continue the installation process.
   - If you are modifying an existing PAS installation, return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes**.

After configuration, PAS allows Docker images to pass through the specified IP address without checking certificates.

### Configure PCF to Access Proxies for Docker Registries

If you have proxies already set up for Docker registries, you should configure PCF to access your Docker registries through a proxy.

To configure PCF to access a Docker registry through a proxy, do the following:

1. On the Installation Dashboard, navigate to **USERNAME > Settings > Proxy Settings**.

2. On the **Update Proxy Settings** pane, complete one of the following fields:

   - **HTTP proxy**: If you have an HTTP proxy server for your Docker registry, enter its IP address.
   - **HTTPS proxy**: If you have an HTTPS proxy server for your Docker registry, enter its IP address.
   - **No proxy**: If you do not use a proxy server, enter the IP address for the Docker registry. This field may already contain proxy settings for the BOSH Director.

   Enter multiple IP addresses as a comma-separated list.
3. Click Update.

4. Return to the Ops Manager dashboard, click Review Pending Changes, and click Apply Changes.
Configuring Cell Disk Cleanup Scheduling

Page last updated:

This topic describes how to configure disk cleanup scheduling on Diego cells in Pivotal Cloud Foundry (PCF).

What is Disk Cleanup

PCF isolates application instances (AIs) from each other using containers that run inside Diego cells. Containers enforce a set of isolation layers including file system isolation. A PCF container file system can either be a PCF stack or the result of pulling a Docker image.

For performance reasons, the cells cache the Docker image layers and the PCF stacks used by running AIs. When PCF destroys an AI or reschedules an AI to a different cell, a chance exist that certain Docker image layers or an old PCF stack becomes unused. If PCF does not clean these unused layers, the cell ephemeral disk will slowly fill.

Disk cleanup is the process of removing unused layers from the cell disk. The disk cleanup process removes all unused Docker image layers and old PCF Stacks, regardless of their size or age.

To perform a detailed analysis of disk usage in your PAS deployment, see Examining GrootFS Disk Usage.

Options for Disk Cleanup

PCF provides the following options for scheduling the disk cleanup process on Diego cells:

- **Never clean up the Cell disk-space**: Pivotal does not recommend selecting this option for production environments.

- **Routinely clean up the Cell disk-space**: This option makes the cell schedule a disk cleanup whenever a container is created. Running the disk cleanup process this frequently can result in a negative impact on the cell performance.

- **Clean up disk-space once usage fills disk**: This option makes the cell schedule the disk cleanup process only when disk usage encroaches on the space reserved for other jobs.

See the Configure Disk Cleanup Scheduling section of this topic to select one of these options.

Recommendations

To select the best option for disk cleanup, you must consider the workload that the Diego cells run.

For PCF installations that primarily run buildpack-based apps, Pivotal recommends selecting the Routinely clean up Cell disk-space option. The Routinely clean up Cell disk-space option ensures that when a new stack becomes available on a cell, the old stack is dropped immediately from the cache.

For PCF installations that primarily run Docker images, or both Docker images and buildpack-based apps, Pivotal recommends selecting the Clean up disk-space once usage fills disk option.

Calculating a Reserve

By default, the space reserved for other jobs is set to a reasonable value for the majority of PCF deployments. The reserve default value accounts for the most typical deployment case where the only other component that writes to /var/vcap/data is the Diego executor cache.

However, if you are deploying any BOSH add-ons that require space in the ephemeral disk, you may need to increase the reserve.

For more information on disk usage and garbage collection in GrootFS, see GrootFS Disk Usage and Examining GrootFS Disk Usage.

⚠️ warning: Setting the reserve to an excessive value can cause garbage collection to occur more frequently. An increase in frequency can result in regular slowdowns in performance of apps on particular Diego Cells. Pivotal recommends leaving the default value for the reserve unless you are advised to modify it.
Configure Disk Cleanup Scheduling

1. Navigate to the PCF Operations Manager Installation Dashboard.

2. Click the Pivotal Application Service (PAS) tile, and navigate to the Application Containers tab.

3. Under Docker Images Disk-Cleanup Scheduling on Cell VMs, select an option.

4. If you select Clean up disk-space once usage fills disk, you can modify the Reserved amount of Disk (MB) field. Pivotal recommends that you leave the default value of 15360 MB, or 15GB, in Reserved amount of Disk (MB) unless you are instructed to change it. For more information, see Calculating a Reserve.

5. Click Save.

Next Steps

If you are configuring PAS for the first time, return to your specific IaaS configuration to continue the installation process.

If you are modifying an existing PAS installation, return to the PCF Ops Manager Installation Dashboard, click Review Pending Changes, and click Apply Changes.
Custom Branding Apps Manager

This topic describes how Pivotal Cloud Foundry operators can visually brand Apps Manager by changing certain text, colors, and images of the interface. Developers view the customized interface when logging in, creating an account, resetting a password, or using Apps Manager.

Operators customize Apps Manager by configuring the Custom Branding and Apps Manager Config pages of the Pivotal Application Service (PAS) tile.

Custom Branding Page

1. In a browser, navigate to the fully qualified domain name (FQDN) of your Ops Manager and log in.
2. Click Pivotal Application Service.
3. Click the Custom Branding tab.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Name</td>
<td>Option to set a custom company name. Defaults to &quot;Pivotal&quot;.</td>
</tr>
<tr>
<td>Accent Color</td>
<td>Input a hexadecimal color code like '#FF0000'.</td>
</tr>
<tr>
<td>Main Logo (PNGs only)</td>
<td>Enter a base64-encoded PNG image string, leave out the mime-type (data:image/png;base64) string. Only enter the base64 encoded data.</td>
</tr>
<tr>
<td>Square Logo (PNGs only)</td>
<td>Same as Main Logo.</td>
</tr>
<tr>
<td>Favicon (PNGs only)</td>
<td>Same as Main Logo.</td>
</tr>
<tr>
<td>Footer Text</td>
<td>Enter a custom footer text. Defaults to &quot;Pivotal Software Inc. All rights reserved.&quot;</td>
</tr>
<tr>
<td>Footer Links</td>
<td>You may configure up to three links in the Apps Manager footer.</td>
</tr>
<tr>
<td>Classification Header/Footer Background Color</td>
<td>Enter a hexadecimal color code like '#FF0000'.</td>
</tr>
<tr>
<td>Classification Header/Footer Text Color</td>
<td>Enter a hexadecimal color code like '#FF0000'.</td>
</tr>
<tr>
<td>Classification Header Content</td>
<td>Enter plain text or HTML markup.</td>
</tr>
<tr>
<td>Classification Footer Content</td>
<td>Enter plain text or HTML markup.</td>
</tr>
</tbody>
</table>
For **Company Name**, enter the name of your organization. If left blank, the name defaults to **Pivotal**.

For **Accent Color**, enter the hexadecimal code for the color used to accent various visual elements, such as the currently selected space in the sidebar. For example, `#71ffda`.

For **Main Logo**, enter a Base64-encoded URL string for a PNG image to use as your main logo. The image can be square or wide. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the Pivotal Logo.

For **Square Logo**, enter a Base64-encoded URL string for a PNG image to use in the Apps Manager header and in places that require a smaller logo. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the Pivotal Logo.

For **Favicon**, enter a Base64-encoded URL string for a PNG image to use as your favicon. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the Pivotal Logo.

For **Footer Text**, enter a string to be displayed as the footer. If left blank, the footer text defaults to **Pivotal Software Inc. All rights reserved.**

To add up to three footer links that appear to the right of the footer text, complete the following steps:

- Click **Add**.
- For **Link text**, enter a label for the link.
- For **Url**, enter an external or relative URL. For example, `http://docs.pivotal.io` or `/tools.html`.

For special notification purposes such as governmental or restricted usage, use the Classification fields to create a special Header and Footer. Enter values in the following fields:

- For **Classification Header/Footer Background Color**, enter the hexadecimal code for the desired background color of the header and footer.
- For **Classification Header/Footer Text Color**, enter the hexadecimal code for the desired color of header and footer text.
- For **Classification Header Content**, enter content for the header in either plain text or HTML. If you enter HTML content, eliminate white spaces and new lines. If you do not provide any content, the custom header will not appear.
- For **Classification Footer Content**, enter content for the footer in either plain text or HTML. If you enter HTML content, eliminate white spaces and new lines. If you do not provide any content, the custom footer will not appear. The Classification footer appears below the normal footer, which you can customize in the **Footer Text** and **Footer Links** fields.

**Note:** The Header and Footer do not appear on the **User Account and Authentication (UAA)** login page.

**Apps Manager Config Page**

1. In a browser, navigate to the fully qualified domain name (FQDN) of your Ops Manager and log in.
2. Click **Pivotal Application Service**.
3. Click the **Apps Manager Config** tab.
4. For **Product Name**, enter text to replace **Apps Manager** in the header and the title of Apps Manager. This text defaults to **Apps Manager** if left blank.

5. For **Marketplace Name**, enter text to replace the header in the Marketplace pages. This text defaults to **Marketplace** if left blank.

6. By default, Apps Manager includes three sidebar links: **Marketplace**, **Docs**, and **Tools**. You can edit existing sidebar links by clicking the name of the link and editing the **Link text** and **Url** fields. Or, you can remove the link by clicking the trash icon next to its name. If you want to add a new sidebar link, click **Add** and complete the **Link text** and **Url** fields.

   **Note:** Removing any of the default links will remove them from the sidebar for all users.

7. (Optional) Enter your desired **Apps Manager Memory Usage** in MBs. The minimum number you can enter is **128**.
8. (Optional) Enter your desired Invitations Memory Usage in MBs. This is the memory limit used to deploy the Invitations app. The minimum number you can enter is \(256\).

9. The Apps Manager Polling Interval field provides a temporary fix if Apps Manager usage degrades Cloud Controller response times. In this case, you can use this field to reduce the load on the Cloud Controller and ensure Apps Manager remains available while you troubleshoot the Cloud Controller. Pivotal recommends that you do not keep this field modified as a long term fix because it can degrade Apps Manager performance. You can optionally do the following:

- Increase the polling interval above the default of \(30\) seconds.

  Note: If you enter a value between \(0\) and \(30\), the field is automatically set to \(30\).

- Disable polling by entering \(0\). This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.

10. The App Details Polling Interval field provides an additional way to reduce the load on the Cloud Controller when the Apps Manager Polling Interval field is not sufficient. This field controls the rate at which Apps Manager polls for data when a user views the Overview page of an app. Pivotal recommends that you do not keep this field modified as a long term fix because it can degrade Apps Manager performance. You can optionally do the following:

- Increase the polling interval above the default of \(10\) seconds.

  Note: If you enter a value between \(0\) and \(30\), the field is automatically set to \(10\).

- Disable polling by entering \(0\). This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.
Reporting App, Task, and Service Instance Usage

Page last updated:

This topic describes how to use the Cloud Foundry Command Line Interface (cf CLI) to retrieve historical system- and org-level usage information about your apps, tasks, and service instances through the Cloud Controller and Usage Service APIs.

Usage reports are compiled from the `/v2/app_usage_events` endpoint. For more information, see List App Usage Events in the Cloud Foundry API documentation.

You can also access usage information by using Apps Manager. For more information, see Reporting Instance Usage with Apps Manager topic.

To retrieve current app and event information from the Cloud Controller, see Retrieving App and Event Information in the open-source Cloud Foundry documentation.

Prerequisite

Before performing the procedures in this topic, ensure that your user is a member of the `cloud_controller.admin` or `usage_service.audit` group. For example, a UAAC admin can add a user to the `usage_service.audit` group with the following command:

```
uaac member add usage_service.audit USERNAME
```

Obtain System Usage Information

You can obtain the following system-wide usage information:

- **App Usage**
- **Task Usage**
- **Service Usage**
- **All Orgs App Usage**

App Usage

Use `curl` to make a request to `/system_report/app_usages` on the Usage Service endpoint to show system-wide app usage data:

```
curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/app_usages" -d {} -X GET \
-H "authorization: cf oauth-token" \
-F "report_time": "2017-04-11 22:28:24 UTC", \
-F "monthly_reports": [ \
  { \
    "month": 4, \
    "year": 2017, \
    "average_app_instances": 17855.256153713308, \
    "maximum_app_instances": 18145, \
    "app_instance_hours": 4686533.080277303 \
  } \
], \
-F "yearly_reports": [ \
  { \
    "year": 2017, \
    "average_app_instances": 16184.9, \
    "maximum_app_instances": 18145, \
    "app_instance_hours": 39207433.0802773 \
  } \
]
```

The following table describes the data generated by this report:

<table>
<thead>
<tr>
<th>Value</th>
<th>Calculation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The total number of hours app instances have been running on the foundation for the month or</td>
</tr>
</tbody>
</table>
Task Usage

Use `curl` to make a request to `/system_report/task_usages` on the Usage Service endpoint to show system-wide task usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/task_usages" -k -v \
-H "authorization: cf oauth-token"
{
  "monthly_reports": [
    {
      "month": 4,
      "year": 2017,
      "total_task_runs": 235,
      "maximum_concurrent_tasks": 7,
      "task_hours": "43045.201944444445"
    }
  ],
  "yearly_reports": [
    {
      "year": 2017,
      "total_task_runs": 2894,
      "maximum_concurrent_tasks": 184,
      "task_hours": "45361.26694444445"
    }
  ]
}
```

Service Usage

Use `curl` to make a request to `/system_report/service_usages` on the Usage Service endpoint to show system-wide service usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/service_usages" -k -v \
-H "authorization: cf oauth-token"
{
  "report_time": "2017-05-11 18:29:14 UTC",
  "monthly_service_reports": [
    {
      "service_name": "fake-service-0507f1fd-2340-49a6-9d43-a347a5f5f6be",
      "service_guid": "177dcfde-cd51-4058-bd86-b98015c295f5",
      "usages": [
        {
          "month": 1,
          "year": 2017,
          "duration_in_hours": 0,
          "average_instances": 0,
          "maximum_instances": 0
        },
        {
          "month": 2,
          "year": 2017,
          "duration_in_hours": 0,
          "average_instances": 0,
          "maximum_instances": 0
        },
        {
          "month": 3,
          "year": 2017,
          "duration_in_hours": "4.182222222222227",
          "average_instances": 0,
          "maximum_instances": 2
        }
      ]
    }
  ]
}
```
All Orgs Usage (Beta)

The `/app_usages` endpoint directly under your system domain returns system-wide app usage for all orgs that you have access to. The command works like the org-specific App Usage command, except you do not supply an org GUID. You can set `order_by` to either `total_instance_hours` or `megabyte_hours`.

The `order_by` ordering defaults to ascending, which lists orgs with the most usage at the end of the output. To change the sort ordering to descending, prepend the `order_by` argument with a `-`, as in `order_by=-megabyte_hours`.

Use curl to make a request to `/app_usages` on the Usage Service endpoint to show system-wide organization usage data:
This beta API endpoint may change; please try it out and help Pivotal make it better.

Obtain Org Usage Information

You can obtain the following org-specific usage information:

- **App Usage**
- **Task Usage**
- **Service Usage**

You must have the GUID of the org you want to obtain information about in order to perform the procedures in this section. To retrieve your org GUID, run the `cf org` command:

```
cf org YOUR-ORG --guid
```

**App Usage**

Use `curl` to make a request to `/app_usages` on the Usage Service endpoint to show app usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.
Use `curl` to make a request to `/task_usages` on the Usage Service endpoint to show task usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/YOUR-ORG-GUID/task_usages?start=YYYY-MM-DD&end=YYYY-MM-DD
.k.v
-H "authorization: cf oauth-token"

{  
  "organization_guid": "8f88362f-547c-4158-808b-4605468387d5",
  "period_start": "2014-01-01",
  "period_end": "2017-04-04",
  "spaces": 
    [ 
      {  
        "space_guid": "e6445eb3-fdac-4049-bafc-94d1703d5e78",
        "space_name": "smoketest",
        "task_summaries": 
          [ 
            {  
              "parent_application_guid": "04cd29d5-1f9e-4900-ac13-2e903f6582a9",
              "parent_application_name": "task-dummy-app",
              "memory_in_mb_per_instance": 256,
              "task_count_for_range": 256,
              "task_count_for_range": 4084,
              "concurrent_task_count_for_range": 5,
              "total_duration_in_seconds_for_range": 37651415
            }
          ]
        },
        {  
          "space_guid": "6dd6665e-877f-4482-acf1-89d07ba9c6ed",
          "space_name": "smoketest-experimental",
          "task_summaries": 
            [ 
              {  
                "parent_application_guid": "5d0411b689-4a27-44cc-91d3-197434f6bed9",
                "parent_application_name": "console-blue",
                "memory_in_mb_per_instance": 256,
                "task_count_for_range": 14,
                "concurrent_task_count_for_range": 2,
                "total_duration_in_seconds_for_range": 20583
              }
            ]
          }
        ]
      }
    ]
}
```

**Note:** In the `/task_usages` endpoint, `memory_in_mb_per_instance` is the memory of the task.
Service Usage

Use `curl` to make a request to `/service_usages` on the Usage Service endpoint to show service usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```bash
```

Example:

```json
{
   "organization_guid": "8d83362f-587a-4148-806b-4407428887b5",
   "period_start": "2016-06-01T00:00:00Z",
   "period_end": "2016-06-13T23:59:59Z",
   "service_usages": [
      {
         "deleted": false,
         "duration_in_seconds": 1377982.52,
         "service_guid": "02802293-b769-44cc-807f-eee331ba9b2b",
         "service_instance_creation": "2016-01-20T18:48:16.000Z",
         "service_instance_deletion": null,
         "service_instance_guid": "b25b4481-19aa-4504-82c9-f303e7e9ed6e",
         "service_instance_name": "something-usage-db",
         "service_instance_type": "managed_service_instance",
         "service_name": "my-mysql-service",
         "service_plan_guid": "70915a70-7311-4f3e-ab0d-4a7d1d3ef2d9",
         "service_plan_name": "20gb",
         "space_guid": "e6445eb3-fd6c-4049-b4fc-94d1703d5e78",
         "space_name": "outer-space"
      }
   ]
}
```

Example: Autogenerated Database Usage Reports

For security compliance and recordkeeping, some PCF customers create Concourse CI/CD pipelines that regularly run JavaScript scripts to do the following:

1. Call the app-usage APIs in the above sections of this topic, as well as the Security Event Logging topic, to collect data about service instances from a PCF foundation. This data includes database instance information such as file or database type, database creator, and date of creation or deletion.

2. Merge and format the data about services instances into a JSON output file and save it to an S3 bucket.

3. Input the file to another process that does the following:
   a. Parses the data
   b. Eliminates duplicate entries from previous reports
   c. Sends the data to a database compliance system through APIs
Reporting Instance Usage with Apps Manager

This topic describes how to retrieve app, task, and service instance usage information using Apps Manager.

You can also retrieve app, task, and service instance usage information using the Usage service API, or the Cloud Foundry API from the Cloud Foundry Command Line Interface (cf CLI). For more information, see the Monitoring App, Task, and Service Instance Usage topic.

There are two ways to monitor app, task, and service instance usage from Apps Manager. The Accounting Report provides a summarized report, and the Usage Report provides a more detailed view of the data.

View the Accounting Report

The Accounting Report displays instance usage information for all orgs in your Pivotal Cloud Foundry (PCF) deployment except the system org. The Accounting Report provides a high-level overview of your usage by displaying your monthly count of app, task, and service instances.

Follow the steps below to access the Accounting Report.

1. Log into the Apps Manager as an admin.
2. From the left navigation of Apps Manager, select Accounting Report.
3. Under App Statistics and Service Usage, view the average and maximum instances in use per month.

Max Concurrent displays the largest number of instances in use at a time over the indicated time period. The Accounting Report calculates these values from the start, stop, and scale app usage events in Cloud Controller.

View the Usage Report

The Usage Report provides a more granular view of your usage by detailing app, task, and service instance usage information for all spaces in a particular org, excluding the system org.

Follow the steps below to access the Usage Report.

1. Log into the Apps Manager as an admin, or as an account with the Org Manager or Org Auditor role. For more information about managing roles, see the Managing User Accounts and Permissions Using the Apps Manager topic.
2. From the dropdown on the left, select the org for which you want to view a usage report.
3. Click **Usage Report** in the upper right.

The top of the Usage Report displays total **App + Task Instance Hours**, **App + Task Memory**, and **Service Instance Hours** by all spaces in the org.

The report provides total usage information for each of your spaces.

To display more detailed information about app, task, and service instance usage in a space, click the name of the space for an expanded view.
Providing a Certificate for Your TLS Termination Point

This topic describes how to configure Transport Layer Security (TLS) termination for HTTP traffic in Pivotal Application Service (PAS) with a TLS certificate, as part of the process of configuring PAS for deployment.

Configure TLS Termination

When you deploy PCF, you must configure the TLS termination for HTTP traffic in your PAS configuration. You can terminate TLS at all of the following points:

- Load Balancer
- Load Balancer and Gorouter
- Gorouter

Follow the guidance in Securing Traffic into Cloud Foundry to choose and configure the TLS termination option for your deployment.

Obtain TLS Certificates

To secure traffic into PCF, you must obtain at least one TLS certificate. See Certificate Requirements for general certificate requirements for deploying PCF.

See the following sections for additional IaaS-specific certificate requirements:

- AWS: Certificate Requirements on AWS
- GCP: Certificate Requirements on GCP

Creating a Wildcard Certificate for PCF Deployments

This section describes how to create or generate a certificate for your PAS environment. If you are deploying to a production environment, you should obtain a certificate from a trusted authority (CA).

For internal development or testing environments, you have two options for creating a required TLS certificates.

- You can create a self-signed certificate, or
- You can have PAS generate the certificate for you.

To create a certificate, you can use a wide variety of tools including OpenSSL, Java’s keytool, Adobe Reader, and Apple’s Keychain to generate a Certificate Signing Request (CSR).

In either case for either self-signed or trusted single certificates, apply the following rules when creating the CSR:

- Specify your registered wildcard domain as the Common Name. For example, *.yourdomain.com.
- If you are using a split domain setup that separates the domains for apps and system components (recommended), then enter the following values in the Subject Alternative Name of the certificate:
  - *.apps.yourdomain.com
  - *.system.yourdomain.com
  - *.login.system.yourdomain.com
  - *.uaa.system.yourdomain.com
- If you are using a single domain setup, then use the following values as the Subject Alternative Name of the certificate:
  - *.login.yourdomain.com

Note: If you are using HAProxy in a PCF deployment, you can choose to terminate SSL/TLS at HAProxy in addition to any of the SSL/TLS termination options above. For more information, see Configuring SSL/TLS Termination at HAProxy.
Generating a RSA Certificate in PAS

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the **Pivotal Application Service** tile in the Installation Dashboard.

3. Click **Networking**.

4. Click **Generate RSA Certificate** to populate the **Certificate and Private Key for HAProxy and Router** fields with RSA certificate and private key information.

5. If you are using a split domain setup that separates the domains for **apps** and **system** components (recommended), then enter the following domains for the certificate:

   ```
   *.yourdomain.com
   *.apps.yourdomain.com
   *.system.yourdomain.com
   *.login.system.yourdomain.com
   *.uaa.system.yourdomain.com
   ```

   For example, `*.example.com`, `*.apps.example.com`, `*.system.example.com`, `*.login.system.example.com`, `*.uaa.system.example.com`.

   *Note: TLS certificates generated for wildcard DNS records only work for a single domain name component or component fragment. For example, a certificate generated for `*.EXAMPLE.com` does not work for `*.apps.EXAMPLE.com` and `*.system.EXAMPLE.com`. The certificate must have both `*.apps.EXAMPLE.com` and `*.system.EXAMPLE.com` attributed to it.*
Enabling Volume Services

This topic describes how Pivotal Cloud Foundry (PCF) operators can deploy NFS or SMB volume services.

Overview

A volume service gives apps access to a persistent filesystem, such as NFS or SMB. By performing the procedures in this topic, operators can add a volume service to the Marketplace that provides an NFS filesystem or an SMB share.

Developers can then use the Cloud Foundry Command Line Interface (cf CLI) to create service instances of the volume service and bind them to their apps. For more information, see the Using an External File System (Volume Services) topic.

Enable NFS Volume Services

To enable NFS volume services in PCF, do the following:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Application Service tile.
3. Click Application Containers.
4. Under Enabling NFS volume services, select Enable.

Note: In a clean install, NFS volume services are enabled by default. In an upgrade, NFS volume services match the setting of the previous deployment.
5. (Optional) To configure LDAP for NFS volume services, perform the following steps:

   - For **LDAP Service Account User**, enter either the full domain name (DN) for the service account or the username of the service account that manages volume services, depending on how your LDAP server is configured. This value needs to be exactly what you would enter when binding the account to your LDAP server.
   - For **LDAP Service Account Password**, enter the password for the service account.
   - For **LDAP Server Host**, enter the hostname or IP address of the LDAP server.
   - For **LDAP Server Port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
   - For **LDAP User Search Base**, enter the location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name. For example, a domain named `cloud.example.com` typically uses the following LDAP User Search Base: `ou=Users,dc=example,dc=com`.
   - For **LDAP Server CA Cert**, you can optionally enter a certificate if your LDAP server supports TLS and you want to enable TLS connections from the NFS driver to your LDAP server. Paste in the root certificate from your CA certificate or your self-signed certificate.

6. Click **Save**.

7. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes** to redeploy.

8. Using the cf CLI, enable access to the service:

   ```
   $ cf enable-service-access nfs
   ```

   To limit access to a specific org, use the `-o` flag, followed by the name of the org where you want to enable access. For more information, see the [Access Control](#) topic.

9. (Optional) Enable access to the `nfs-legacy` service. See [NFS Volume Service](#) for details about the differences between the two `nfs` services.
After completing these steps, developers can use the `cf` CLI to create service instances of the `nfs` service and bind them to their apps.

**NFS Security**

You can use ASGs and LDAP to secure your NFS server against traffic apps running on PCF:

- **Application Security Groups (ASGs)**—Prevent apps from sending traffic directly to your NFS ports. Apps should never need to use NFS ports directly. Pivotal recommends defining an ASG that blocks direct access to your NFS server IP, especially ports 111 and 2049. For more information on setting up ASGs, see Application Security Groups.

- **LDAP**—In addition to ASGs, secures the NFS volume service so that app developers cannot bind to the service using an arbitrary UID. App developers also cannot gain access to sensitive data. With LDAP support enabled, app developers must provide credentials for any user they wish to bind as.

The Diego cells running on PCF must be able to reach your LDAP server on the port you use for connections, which are typically 389 or 636. You cannot limit which Diego cells have access to your NFS or LDAP servers.

**Enable SMB Volume Services**

To enable SMB volume services in PCF, do the following:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the Pivotal Application Service tile.
3. Select the Application Containers pane.
4. Check the box next to Enable SMB volume services.
5. Click Save.
6. Select the Errands pane.
7. Set the SMB Broker Errand to On.
8. Click Save.
9. Return to the Ops Manager Installation Dashboard, click Review Pending Changes, and click Apply Changes to redeploy.
10. Using the `cf` CLI, enable access to the service:

```
cf enable-service-access smb
```

To limit access to a specific org, use the `-o` flag, followed by the name of the org where you want to enable access. For more information, see the Access Control topic.

After completing these steps, developers can use the `cf` CLI to create service instances of the `smb` service and bind them to their apps.
Rotating Runtime CredHub Encryption Keys

This topic discusses how to rotate runtime CredHub encryption keys. Encryption keys are values that CredHub uses to obscure stored secrets. When an operator marks an additional key as primary, CredHub can rotate in that additional key as the encryption key.

During this credential rotation process, the initial encryption key is used to access the hidden value. That value is then stored again by the additional encryption key.

⚠️ **warning:** If you remove an encryption key and click *Apply Changes* before the rotation completes, the deployment breaks. If this happens, you can no longer access data stored with the deleted key.

**Rotate PAS Encryption Keys**

To rotate PAS encryption keys, do the following:

1. Navigate to the *Ops Manager Installation Dashboard*.
2. Click the *Pivotal Application Service* tile.
3. Select the *CredHub* tab.
4. In the *Encryption Keys* section, click *Add*.
5. For *Name*, enter the name of your new encryption key.
6. For *Key*, enter your new encryption key.
7. Select the *Primary* check box.
8. Click *Save*.
9. Navigate to the *Ops Manager Installation Dashboard*.
10. Click *Review Pending Changes*, then *Apply Changes*.

**Verify PAS Encryption Key Rotation**

Follow the steps below to verify that the rotation completes.

1. Click the *Pivotal Application Service* tile.
2. Select the *Status* tab.
3. Within the *CredHub* job, locate *Index 0*.
4. Within the Logs column, click the correlating download icon.

5. Select the Logs tab.

6. Click the corresponding link to retrieve the downloaded log file.

7. Unzip the log file.

8. Unzip the larger of the two nested directories.

9. Ops Manager generates a compressed file for each CredHub VM that exists on your deployment. Unzip each of these compressed files.

10. Open the credhub directory.

11. Open the credhub.log file. If the PAS credential rotation completed successfully, the CredHub log contains the following string: Successfully rotated NUMBER-OF-CREDENTIALS items.

12. Remove the old encryption key.

13. Click the trashcan icon that corresponds to the old encryption key.

14. Click Save.

15. Navigate to the Ops Manager Installation Dashboard.

16. Click Review Pending Changes, then Apply Changes.
Administrating PAS

This guide provides instructions for administrating Pivotal Application Service (PAS).

See the following topics:

- Managing the Runtime
- User Accounts and Communications
- Routing
- Isolation Segments
Managing Custom Buildpacks

Page last updated:

This topic describes how an admin can manage additional buildpacks in Cloud Foundry using the Cloud Foundry Command Line Interface tool (cf CLI). If your application uses a language or framework that the Cloud Foundry system buildpacks do not support, you can take one of the following actions:

- Write your own buildpack
- Customize an existing buildpack
- Use a Cloud Foundry Community Buildpack
- Use a Heroku Third-Party Buildpack

Add a Buildpack

Note: You must be a Cloud Foundry admin user to run the commands discussed in this section.

To add a buildpack, run the `cf create-buildpack BUILDPACK PATH POSITION [--enable|--disable]` command. The arguments to `cf create-buildpack` specify the following:

- **BUILDPACK** specifies the buildpack name.
- **PATH** specifies the location of the buildpack. PATH can point to a zip file, the URL of a zip file, or a local directory.
- **POSITION** specifies where to place the buildpack in the detection priority list. For more information, see the [Buildpack Detection](#) topic.
- **enable** or **disable** specifies whether to allow apps to be pushed with the buildpack. This argument is optional, and defaults to enable. When a buildpack is disabled, app developers cannot push apps using that buildpack.

To confirm that you have successfully added a buildpack, run `cf buildpacks`.

The following example shows the output from running the `cf buildpacks` command after an administrator adds a Python buildpack:

```
$ cf buildpacks
Getting buildpacks...

<table>
<thead>
<tr>
<th>buildpack</th>
<th>position</th>
<th>enabled</th>
<th>locked</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruby_buildpack</td>
<td>1</td>
<td>true</td>
<td>false</td>
<td>buildpack_ruby_v46-245-g264a8.zip</td>
</tr>
<tr>
<td>nodejs_buildpack</td>
<td>2</td>
<td>true</td>
<td>false</td>
<td>buildpack_nodejs_v8-177-g2885c8.zip</td>
</tr>
<tr>
<td>java_buildpack</td>
<td>3</td>
<td>true</td>
<td>false</td>
<td>buildpack_java_v2.1.zip</td>
</tr>
<tr>
<td>python_buildpack</td>
<td>4</td>
<td>true</td>
<td>false</td>
<td>buildpack_python_v2.7.6.zip</td>
</tr>
</tbody>
</table>
```

Rename a Buildpack

To rename a buildpack, run `cf rename-buildpack BUILDPACK-NAME NEW-BUILDPACK-NAME`. Replace **BUILDPACK-NAME** with the original buildpack name, and **NEW-BUILDPACK-NAME** with the new buildpack name.

For more information about renaming buildpack, see the [cf CLI documentation](#).

Update a Buildpack

```
```

For more information about updating buildpacks, see the [cf CLI documentation](#).

Delete a Buildpack

Note: You must be a Cloud Foundry admin user to run the commands discussed in this section.

© Copyright Pivotal Software Inc, 2013-2019
Lock and Unlock a Buildpack

Every new version of Cloud Foundry includes an updated buildpack. By default, your deployment applies the most recent buildpack when you upgrade. In some cases, however, you may want to preserve an existing buildpack, rather than upgrade to the latest version. For example, if an app you deploy depends on a specific component in Buildpack A that is not available in Buildpack B, you may want to continue using Buildpack A.

The `--lock` flag lets you continue to use your existing buildpack even after you upgrade. Locked buildpacks are not updated when PCF updates. You must manually unlock them to update them.

If you elect to use the `--unlock` flag, your deployment will apply the most recent buildpack when you upgrade PCF.


This feature is also available through the API. For more information, see Lock or unlock a Buildpack in the Cloud Foundry API documentation.

Disabling Custom Buildpacks

You can disable custom buildpacks using your Ops Manager PAS tile. From the Cloud Controller tab, check the Disable Custom Buildpacks checkbox, as shown in the image below.

By default, the cf CLI gives developers the option of using a custom buildpack when they deploy apps to PAS. To do so, they use the `--` option to provide a custom buildpack URL with the `cf push` command. The Disable Custom Buildpacks checkbox prevents the `--` option from being used with external buildpack URLs.

For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.
Using Docker in Cloud Foundry

Page last updated:

This topic describes how Cloud Foundry (CF) operators can enable CF developers to run their apps in Docker containers, and explains how Docker works in Cloud Foundry.

For information about Diego, the Cloud Foundry component that manages application containers, see the Diego Architecture topic. For information about how CF developers push apps with Docker images, see Deploying an App with Docker.

Enable Docker

By default, apps deployed with the `cf push` command run in standard Cloud Foundry Linux containers. With Docker support enabled, Cloud Foundry can also deploy and manage apps running in Docker containers.

To deploy apps to Docker, developers run `cf push` with the `--docker-image` option and the location of a Docker image to create the containers from. See the Push a Docker Image topic for information about how CF developers push apps with Docker images.

To enable Docker support on a CF deployment, an operator must do the following:

- Enable the `diego_docker` feature flag.
- Configure access to any Docker registries that developers want to use images from.

Enable and Disable the diego_docker Feature Flag

The `diego_docker` feature flag governs whether a CF deployment supports Docker containers.

To enable Docker support, run:

```
$ cf enable-feature-flag diego_docker
```

To disable Docker support, run:

```
$ cf disable-feature-flag diego_docker
```

Note: Disabling the `diego_docker` feature flag stops all Docker-based apps in your deployment within a few convergence cycles, on the order of a minute.

Configure Docker Registry Access

To support Docker, Pivotal Cloud Foundry needs the ability to access Docker registries using either a Certificate Authority or an IP address whitelist. The Using Docker Registries topic explains how to configure this access.

Docker Image Contents

A Docker image consists of a collection of layers. Each layer consists of one or both of the following:

- Raw bits to download and mount. These bits form the file system.
- Metadata that describes commands, users, and environment for the layer. This metadata includes the `ENTRYPOINT` and `CMD` directives, and is specified in the Dockerfile.

How Garden-runC Creates Containers

© Copyright Pivotal Software Inc, 2013-2019
Diego currently uses Garden-runC to construct Linux containers.

Both Docker and Garden-runC use libraries from the Open Container Initiative (OCI) to build Linux containers. After creation, these containers use name space isolation, or namespaces, and control groups, or cgroups, to isolate processes in containers and limit resource usage. These are common kernel resource isolation features used by all Linux containers.

Note: PAS versions v1.8.8 and above use Garden-runC instead of Garden-Linux.

Before Garden-runC creates a Linux container, it creates a file system that is mounted as the root file system of the container. Garden-runC supports mounting Docker images as the root file systems for the containers it creates.

When creating a container, both Docker and Garden-runC perform the following actions:

- Fetch and cache the individual layers associated with a Docker image
- Combine and mount the layers as the root file system

These actions produce a container whose contents exactly match the contents of the associated Docker image.

Earlier versions of Diego used Garden-Linux. For more information, see the Garden topic.

How Diego Runs and Monitors Processes

After Garden-runC creates a container, Diego runs and monitors the processes inside of it.

To determine which processes to run, the Cloud Controller fetches and stores the metadata associated with the Docker image. The Cloud Controller uses this metadata to perform the following actions:

- Runs the start command as the user specified in the Docker image
- Instructs Diego and the Gorouter to route traffic to the lowest-numbered port exposed in the Docker image, or port 8080 if the Dockerfile does not explicitly expose a listen port.

Note: When launching an application on Diego, the Cloud Controller honors any user-specified overrides such as a custom start command or custom environment variables.

Docker Security Concerns in a Multi-Tenant Environment

The attack surface area for a Docker-based container running on Diego remains somewhat higher than that of a buildpack application because Docker allows users to fully specify the contents of their root file systems. A buildpack application runs on a trusted root filesystem.

Garden-runC provides features that allow the platform to run Docker images more securely in a multi-tenant context. In particular, Cloud Foundry uses the user-namespacing feature found on modern Linux kernels to ensure that users cannot gain escalated privileges on the host even if they escalate privileges within a container.

The Cloud Controller always runs Docker containers on Diego with user namespaces enabled. This security restriction prevents certain features, such as the ability to mount FuseFS devices, from working in Docker containers. Docker applications can use fuse mounts through volume services, but they cannot directly mount fuse devices from within the container.

To mitigate security concerns, Cloud Foundry recommends that you run only trusted Docker containers on the platform. By default, the Cloud Controller does not allow Docker-based applications to run on the platform.
Creating and Managing Users with the cf CLI

Using the Cloud Foundry Command Line Interface (cf CLI), administrators, Org Managers, and Space Managers can manage users. Cloud Foundry uses role-based access control, with each role granting permissions in either an organization or an application space.

For more information, see Organizations, Spaces, Roles, and Permissions.

About Roles

To manage all users, organizations, and roles with the cf CLI, log in with your admin credentials. In Pivotal Operations Manager, refer to PAS > Credentials for the admin name and password.

If the feature flag `set_roles_by_username` is enabled, Org Managers can assign org roles to existing users in their org and Space Managers can assign space roles to existing users in their space. For more information about using feature flags, see the Feature Flags topic.

Creating and Deleting Users

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>COMMAND</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new user</td>
<td>cf create-user USERNAME PASSWORD</td>
<td>cf create-user Alice pa55w0rd</td>
</tr>
<tr>
<td>Create a new user, specifying LDAP as an external identity provider</td>
<td>cf create-user USERNAME -origin ORIGIN</td>
<td>cf create-user Aayah ldap</td>
</tr>
<tr>
<td>Create a new user, specifying SAML or OpenID Connect as an external identity provider</td>
<td>cf create-user USERNAME -origin ORIGIN</td>
<td>cf create-user Alko provider-alias</td>
</tr>
<tr>
<td>Delete a user</td>
<td>cf delete-user USERNAME</td>
<td>cf delete-user Alice</td>
</tr>
</tbody>
</table>

Creating Administrator Accounts

To create a new administrator account, use the UAA CLI.

Note: The cf CLI cannot create new administrator accounts.

Org and App Space Roles

A user can have one or more roles. The combination of these roles defines the user’s overall permissions in the org and within specific app spaces in that org.

Org Roles

Valid org roles are OrgManager, BillingManager, and OrgAuditor.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>COMMAND</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the organizations belonging to an account</td>
<td>cf orgs</td>
<td>cf orgs</td>
</tr>
<tr>
<td>View all users in an organization by role</td>
<td>cf org-users ORGANIZATION-NAME</td>
<td>cf org-users my-example-org</td>
</tr>
<tr>
<td>Assign an org role to a user</td>
<td>cf set-org-role USERNAME ORGANIZATION-NAME ROLE</td>
<td>cf set-org-role Alice my-example-org OrgManager</td>
</tr>
<tr>
<td>Remove an org role from a user</td>
<td>cf unset-org-role USERNAME ORGANIZATION-NAME ROLE</td>
<td>cf unset-org-role Alice my-example-org OrgManager</td>
</tr>
</tbody>
</table>

If multiple accounts share a username, `set-org-role` and `unset-org-role` return an error. See Identical Usernames in Multiple Origins for details.
App Space Roles

Each app space role applies to a specific app space.

Valid app space roles are SpaceManager, SpaceDeveloper, and SpaceAuditor.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>COMMAND</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the spaces in an org</td>
<td>cf spaces</td>
<td>cf spaces</td>
</tr>
<tr>
<td>View all users in a space by role</td>
<td>cf space-users ORGANIZATION-NAME SPACE-NAME</td>
<td>cf space-users my-example-org development</td>
</tr>
<tr>
<td>Assign a space role to a user</td>
<td>cf set-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE</td>
<td>cf set-space-role Alice my-example-org development SpaceAuditor</td>
</tr>
<tr>
<td>Remove a space role from a user</td>
<td>cf unset-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE</td>
<td>cf unset-space-role Alice my-example-org development SpaceAuditor</td>
</tr>
</tbody>
</table>

If multiple accounts share a username, `set-space` and `unset-space` return an error. See Identical Usernames in Multiple Origins for details.
Creating and Managing Users with the UAA CLI (UAAC)

Using the UAA Command Line Interface (UAAC), an administrator can create users in the User Account and Authentication (UAA) server.

**Note:** The UAA only creates users in UAA, and does not assign roles in the Cloud Controller database (CCDB). In general, administrators create users using the Cloud Foundry Command Line Interface (cf CLI). The cf CLI both creates user records in the UAA and associates them with org and space roles in the CCDB. Before administrators can assign roles to the user, the user must log in through Apps Manager or the cf CLI for the user record to populate the CCDB. Review the Creating and Managing Users with the cf CLI topic for more information.

For additional details and information, refer to the following topics:

- UAA Overview
- UAA Sysadmin Guide
- Other UAA Documentation

**Note:** UAAC requires Ruby v2.3.1 or later. If you have an earlier version of Ruby installed, install v2.3.1 or later before using the UAAC.

For more information about which roles can perform various operations in Cloud Foundry, see the Roles and Permissions topic.

Create an Admin User

1. Install the UAA CLI, `uaac`.

   ```bash
   $ gem install cf-uaac
   ```

2. Use the `uaac target uaa.YOUR-DOMAIN` command to target your UAA server.

   ```bash
   $ uaac target uaa.example.com
   ```

3. Record the `uaa:admin:client_secret` from your deployment manifest.

4. Run `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-SECRET` with the admin secret you have retrieved in previous step. UAAC stores the token in `~/.uaac.yml`.

   ```bash
   $ uaac token client get admin -s MyAdminSecret
   ```

5. Use the `uaac contexts` command to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

   ```bash
   $ uaac contexts
   
   [1]*[admin]
   
   client_id: admin
   access_token: aliudE2jg0hJCKtn3123.e
   token_type: bearer
   expires_in: 43200
   scope: uaa.admin clients.secret scim.read
   jti: 91b3-abcd123
   ```

6. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for `scim.write`. The value `scim.write` represents sufficient permissions to create accounts.

7. If the admin user lacks permissions to create accounts, add the permissions by following these steps:

   - Run `uaac client update admin --authorities 'EXISTING-PERMISSIONS scim.write'` to add the necessary permissions to the admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
   - Run `uaac token delete` to delete the local token.
   - Run `uaac token client get admin` to obtain an updated access token from the UAA server.
Create an Admin Read-Only User

The admin read-only account can view but not modify almost all Cloud Controller API resources. The admin read-only account cannot view process stats or logs.

If you want to create an admin read-only user account, then perform the following steps:

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa:scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-SECRET` with your admin secret. UAAC stores the token in `~/.uaac.yml`.

3. Run the following command to create an admin read-only user: `uaac user add NEW-USERNAME -p NEW-PASSWORD --emails NEW-EMAIL` Replace `NEW-USERNAME`, `NEW-PASSWORD`, and `NEW-EMAIL` with appropriate information.

4. Run `uaac member add GROUP NEW-USERNAME` to add the new admin read-only account to the groups `cloud_controller.admin_read_only` and `scim.read`.

Create a Global Auditor

The global auditor account has read-only access to almost all Cloud Controller API resources but cannot access secret data such as environment variables. The global auditor account cannot view process stats or logs.

Perform the following steps to create a global auditor account:

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa:scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-SECRET` with your admin secret. UAAC stores the token in `~/.uaac.yml`.
Grant Admin Permissions to an External Group (SAML or LDAP)

To grant all users under an external group admin permissions, do the following:

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa:scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-SECRET` with your admin secret. UAAC stores the token in `~/.uaac.yml`.

3. Follow the procedure that corresponds to your use case:
   - Grant Admin Permissions for LDAP
   - Grant Admin Permissions for SAML

   **Note:** The UAA will not grant scopes for users in external groups until the next time the user logs in. This means that users granted scopes from external group mappings must log out from PCF and log back in before their new scope takes effect.

Grant Admin Permissions for LDAP

Run the commands below to grant all users under the mapped LDAP Group admin permissions. Replace `GROUP-DISTINGUISHED-NAME` with an appropriate group name.

1. `uaac group map --name scim.read "GROUP-DISTINGUISHED-NAME"`
2. `uaac group map --name scim.write "GROUP-DISTINGUISHED-NAME"`
3. `uaac group map --name cloud_controller.admin "GROUP-DISTINGUISHED-NAME"`

Grant Admin Permissions for SAML

1. Retrieve the name of your SAML provider by navigating to the PAS tile on the Ops Manager Installation Dashboard, clicking Authentication and Enterprise SSO, and recording the value under Provider Name. For more information about configuring PCF for a SAML identity provider, see the Configuring Authentication and Enterprise SSO for PAS topic.

2. Run the commands below to grant all users under the mapped SAML group admin permissions. Replace `GROUP-NAME` with the group name, and `SAAML-PROVIDER-NAME` with the name of your SAML provider.

   a. `uaac group map --name scim.read "GROUP-NAME" --origin SAAML-PROVIDER-NAME`
   b. `uaac group map --name scim.write "GROUP-NAME" --origin SAAML-PROVIDER-NAME`
   c. `uaac group map --name cloud_controller.admin "GROUP-NAME" --origin SAAML-PROVIDER-NAME`
Create Users

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa:scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `cf login -u NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD` to log in.

```bash
$ cf login -u Adam -p newAdminSecretPassword
```

3. Run `cf create-user NEW-USER-NAME NEW-USER-PASSWORD` to create a new user.

```bash
$ cf create-user Charlie aNewPassword
```

Change Passwords

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa:scim` section of your deployment manifest for the user name and password of an admin user.

2. Run `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-SECRET` with your admin secret. UAAC stores the token in `~/.uaac.yml`.

```bash
$ uaac token client get admin -s MyAdminSecret
```

3. Run `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```bash
$ uaac contexts
```

4. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for `password.write`. The value `password.write` represents sufficient permissions to change passwords.

5. If the admin user lacks permissions to change passwords, add the permissions by following these steps:
   - Run `uaac client update admin --scope "EXISTING-PERMISSIONS password.write"` to add the necessary permissions to the admin user account on the UAA server. Replace `EXISTING-PERMISSIONS` with the current contents of the `scope` section from `uaac contexts`.
   - Run `uaac token delete` to delete the local token.
   - Run `uaac token client get admin` to obtain an updated access token from the UAA server.

```bash
$ uaac contexts
```

6. Run `uaac password set USER-NAME -p TEMP-PASSWORD` to change an existing user password to a temporary password.

```bash
$ uaac password set Charlie -p ThidsATempPassword
```

7. Provide the `TEMP-PASSWORD` to the user. Have the user use `cf target api.YOUR-DOMAIN`, `cf login -u USER-NAME -p TEMP-PASSWORD`, and `cf passwd` to change the temporary password. See the Configuring UAA Password Policy topic to configure the password policy.
Retrieve User Email Addresses

Some Cloud Foundry components, like Cloud Controller, only use GUIDs for user identification. You can use the UAA to retrieve the emails of your Cloud Foundry instance users either as a list or, for a specific user, with that user’s GUID.

Follow the steps below to retrieve user email addresses:

1. Run `uaac target uaa.YOUR-DOMAIN` to target your UAA server.

```bash
$ uaac target uaa.example.com
```

2. Record the `uaa:admin:client_secret` from your deployment manifest.

3. Run `uaac token client get admin --client-secret ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. Replace `ADMIN-CLIENT-SECRET` with your admin secret. UAAC stores the token in `~/.uaac.yml`.

```bash
$ uaac token client get admin --client-secret MyAdminSecret
```

4. Run `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```bash
$ uaac contexts
```

5. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for `scim.read`. The value `scim.read` represents sufficient permissions to query the UAA server for user information.

6. If the admin user lacks permissions to query the UAA server for user information, add the permissions by following these steps:

   - Run `uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"` to add the necessary permissions to the admin user account on the UAA server. Replace `EXISTING-PERMISSIONS` with the current contents of the `scope` section from `uaac contexts`.
   - Run `uaac token delete` to delete the local token.
   - Run `uaac token client get admin` to obtain an updated access token from the UAA server.

```bash
$ uaac contexts
```

7. Run `uaac users` to list your Cloud Foundry instance users. By default, the `uaac users` command returns information about each user account including GUID, name, permission groups, activity status, and metadata. Use the `--attributes emails` or `-a emails` flag to limit the output of `uaac users` to email addresses.

© Copyright Pivotal Software Inc, 2013-2019
addresses.

```bash
$ uaac users --attributes emails
resources:
  emails:
    value: user1@example.com
  emails:
    value: user2@example.com
  emails:
    value: user3@example.com

8. Run `uaac users "id eq GUID" --attributes emails` with the GUID of a specific user to retrieve that user’s email address.

```bash
$ uaac users "id eq 'aabbcc11-22a5-87-8056-beaf84'" --attributes emails
resources:
  emails:
    value: user1@example.com
```
Creating and Modifying Quota Plans

Quota plans are named sets of memory, service, and instance usage quotas. For example, one quota plan might allow up to 10 services, 10 routes, and 2 GB of RAM, while another might offer 100 services, 100 routes, and 10 GB of RAM. Quota plans have user-friendly names, but are referenced in Cloud Foundry (CF) internal systems by unique GUIDs.

Quota plans are not directly associated with user accounts. Instead, every org has a list of available quota plans, and the account admin assigns a specific quota plan from the list to the org. Everyone in the org shares the quotas described by the plan. There is no limit to the number of defined quota plans an account can have, but only one plan can be assigned at a time.

You must set a quota plan for an org, but you can choose whether to set a space quota.

Org Quota Plan Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Valid Values</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name you use to identify the plan</td>
<td>A sequence of letters, digits, and underscore characters. Quota plan names within an account must be unique.</td>
<td>silver_quota</td>
</tr>
<tr>
<td>memory_limit</td>
<td>Maximum memory usage allowed</td>
<td>An integer and a unit of measurement like M, MB, G, or GB</td>
<td>2048M</td>
</tr>
<tr>
<td>app_instance_limit</td>
<td>Maximum app instances allowed</td>
<td>An integer</td>
<td>25</td>
</tr>
<tr>
<td>non_basic_services_allowed</td>
<td>Determines whether users can provision instances of non-free service plans</td>
<td>true or false</td>
<td>true</td>
</tr>
<tr>
<td>total_routes</td>
<td>Maximum routes allowed</td>
<td>An integer</td>
<td>500</td>
</tr>
<tr>
<td>total_reserved_route_ports</td>
<td>Maximum routes with reserved ports</td>
<td>An integer not greater than total_routes</td>
<td>60</td>
</tr>
<tr>
<td>total_services</td>
<td>Maximum services allowed</td>
<td>An integer</td>
<td>25</td>
</tr>
<tr>
<td>trial_db_allowed</td>
<td>Legacy Field. Value can be ignored.</td>
<td>true or false</td>
<td>true</td>
</tr>
</tbody>
</table>

Default Quota Plan for an Org

Cloud Foundry installs with a quota plan named default with the following values:

- Memory Limit: 10240 MB
- Total Routes: 1000
- Total Services: 100
- Non-basic Services Allowed: True
- Trial DB Allowed: True

Create a New Quota Plan for an Org

⚠️ Note: The org manager sets and manages quotas. See the Orgs, Spaces, Roles, and Permissions topic for more information.
You must set an org quota. You can create a new quota plan for org with

```
cf create-quota
```

### Use cf create-quota

In a terminal window, run the following command. Replace the placeholder attributes with the values for this quota plan:

```
cf create-quota QUOTA [-m TOTAL-MEMORY] [-i INSTANCE-MEMORY] [-r ROUTES] [-s SERVICE-INSTANCES] [--allow-paid-service-plans]
```

This command accepts the following flags:

- `-m`: Total amount of memory
- `-i`: Maximum amount of memory an application instance can have (`-1` represents an unlimited amount)
- `-r`: Total number of routes
- `-s`: Total number of service instances
- `--allow-paid-service-plans`: Can provision instances of paid service plans

**Example:**

```
$ cf create-quota small -m 2048M -i 1024M -r 10 -s 10 --allow-paid-service-plans
```

### Modify an Existing Quota Plan for an Org

**Use cf update-quota**

1. Run `cf quotas` to find the names of all quota definitions available to your org. Record the name of the quota plan to be modified.

```
$ cf quotas
Getting quotas as admin@example.com...
OK

name       total memory limit instance memory limit routes service instances paid service plans
free       0          0          1000    0       disallowed
paid       10G        0          1000    -1       allowed
small      2G         0          1000    10       allowed
trial      2G         0          1000    10       disallowed
```

2. Run the following command, replacing QUOTA with the name of your quota.

```
```

This command accepts the following flags:

- `-i`: Maximum amount of memory an application instance can have (`-1` represents an unlimited amount)
- `-m`: Total amount of memory a space can have
- `-n`: New name
- `-r`: Total number of routes
- `-s`: Total number of service instances
- `--allow-paid-service-plans`: Can provision instances of paid service plans
- `--disallow-paid-service-plans`: Can not provision instances of paid service plans

**Example:**

```
$ cf update-quota small -i 2048M -m 4096M -r 10 -s 10 --allow-paid-service-plans
```

### Create and Modify Quota Plans for a Space

© Copyright Pivotal Software Inc, 2013-2019 894 2.5
For each org, Org Managers create and modify quota plans for spaces in the org. If an Org Manager allocates a space quota, CF verifies that resources do not exceed the allocated space limit. For example, when a Space Developer deploys an application, CF first checks the memory allocation at the space level, then at the org level.

Perform the following procedures to create and modify quota plans for individual spaces within an org.

Create a New Quota Plan for a Space

In a terminal window, run the following command to create a quota for a space. Replace the placeholder attributes with the values for this quota plan:

```
cf create-space-quota QUOTA [-i INSTANCE-MEMORY] [-m MEMORY] [-r ROUTES] [-s SERVICE-INSTANCES] [--allow-paid-service-plans]
```

Example:

```
$ cf create-space-quota big -i 1024M -m 4096M -r 20 -s 20 --allow-paid-service-plans
```

Modify a Quota Plan for a Space

Run `cf space-quotas` to find the names of all space quota available to your org. Record the name of the quota plan to be modified.

```
cf space-quotas
```

Getting quotas as admin@example.com...

OK

<table>
<thead>
<tr>
<th>name</th>
<th>total memory limit</th>
<th>instance memory limit</th>
<th>routes</th>
<th>service instances</th>
<th>paid service plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td>2G</td>
<td>unlimited</td>
<td>0</td>
<td>10</td>
<td>allowed</td>
</tr>
<tr>
<td>trial</td>
<td>2G</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>allowed</td>
</tr>
</tbody>
</table>

To modify that quota, use the `update-space-quota` command. Replace the placeholder attributes with the values for this quota plan.

```
```

Example:

```
$ cf update-space-quota big -i 20 -m 4096M -n bigger -r 20 -s 20 --allow-paid-service-plans
```

Run cf help

For more information regarding quotas, run `cf help` to view a list and brief description of all cf CLI commands. Scroll to view org and space quotas usage and information.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quotas</td>
<td>List available usage quotas</td>
</tr>
<tr>
<td>quota</td>
<td>Show quota info</td>
</tr>
<tr>
<td>set-quota</td>
<td>Assign a quota to an org</td>
</tr>
<tr>
<td>create-quota</td>
<td>Define a new resource quota</td>
</tr>
<tr>
<td>delete-quota</td>
<td>Delete a quota</td>
</tr>
<tr>
<td>update-quota</td>
<td>Update an existing resource quota</td>
</tr>
<tr>
<td>share-private-domain</td>
<td>Share a private domain with an org</td>
</tr>
<tr>
<td>unshare-private-domain</td>
<td>Unshare a private domain with an org</td>
</tr>
<tr>
<td>space-quotas</td>
<td>List available space resource quotas</td>
</tr>
<tr>
<td>space-quota</td>
<td>Show space quota info</td>
</tr>
<tr>
<td>create-space-quota</td>
<td>Define a new space resource quota</td>
</tr>
<tr>
<td>update-space-quota</td>
<td>Update an existing space quota</td>
</tr>
<tr>
<td>delete-space-quota</td>
<td>Delete a space quota definition and unassign the space quota from all spaces</td>
</tr>
<tr>
<td>set-space-quota</td>
<td>Assign a space quota definition to a space</td>
</tr>
<tr>
<td>unset-space-quota</td>
<td>Unassign a quota from a space</td>
</tr>
</tbody>
</table>
Using Metadata

This topic describes metadata in Pivotal Application Service (PAS) and provides instructions for adding, updating, removing, and viewing metadata.

For additional information about adding metadata with the Cloud Foundry API (CAPI), see the Metadata section in the CAPI documentation.

About Metadata

PAS allows you to add metadata to objects such as spaces and apps. You can use metadata to provide additional information about the objects in your PAS deployment. This can help with operating, monitoring, and auditing.

For example, you can tag objects with metadata that describes the type of environment they belong to. You could also use metadata to describe app characteristics, such as front end or back end. Other examples include billing codes, points of contact, resource consumption, and information about security or risk.

Here is the full list of objects to which you can add metadata using the Cloud Foundry API:

- Apps
- Builds
- Buildpacks
- Deployments
- Droplets
- Isolation Segments
- Orgs
- Packages
- Processes
- Spaces
- Stacks
- Tasks
- Revisions

You can add metadata to apps and spaces using Apps Manager. For more information about adding metadata to apps using Apps Manager, see Managing Apps and Service Instances Using Apps Manager. For more information about adding metadata to spaces using Apps Manager, see Managing Orgs and Spaces Using Apps Manager.

Types of Metadata

You can add two types of metadata to objects in PAS:

- **Labels:** Labels allow you to identify and select PAS objects. For example, if you have labeled all apps running in production, or all spaces that contain Internet-facing apps, you can then search for them.

- **Annotations:** Annotations allow you to add non-identifying metadata to PAS objects. You cannot query based on annotations. Also, there are fewer restrictions for key-value pairs of annotations than there are for labels. For example, you can include contact information of persons responsible for the object, or tool information for debugging purposes.

Metadata Requirements

The following tables describe requirements for creating metadata.

Requirements for Labels

The following table describes the requirements for creating labels.
## Requirements for Annotations

The following table describes the requirements for creating annotations.

<table>
<thead>
<tr>
<th>Annotation Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Part of Annotation</strong></td>
</tr>
<tr>
<td>Key</td>
</tr>
<tr>
<td>Value</td>
</tr>
</tbody>
</table>

## Label Prefixes

You can ensure a label key is easily differentiated from other keys by using a prefix. A prefix is a namespace pattern that helps you more clearly identify objects. Prefixes are in DNS subdomain format: `prefix.example.com`.

Consider an example in which you have two scanner tools: one for security and one for compliance. Both tools use a `scanned` label. You can disambiguate between the two tools using a prefix. The security tool can prefix a label with `security.example.com/scanned` and the compliance tool can prefix a label with `compliance.example.com/scanned`.

## Add Metadata to an Object

The sections below describe how to add labels and annotations to objects.

### Add a Label

The following procedure describes how to add a label:

1. To add a label to an object, run the following command:

   ```bash
cf curl v3/OBJECT-ENDPOINT/GUID 
   -X PATCH 
   -d '{
   "metadata": {
   "labels": {
   "LABEL-KEY": "LABEL-VALUE"
   }
   }
   '```

   Where:

   - `OBJECT-ENDPOINT` is the CAPI endpoint for the type of object you want to label, such as `apps` or `organizations`
   - `GUID` is the GUID of the object you want to label
• **LABEL-KEY** is the key for the label

• **LABEL-VALUE** is the corresponding value for the label key

For example, the following command labels an app as "environment": "production".

```
$ cf curl v3/apps/1cb006ee-fb05-47e1-b541-c34179ddc446 
-X PATCH
-d '{
   "metadata": {
      "labels": {
         "environment": "production"
      }
   }
}'
```

Add an Annotation

The following procedure describes how to add an annotation:

1. To add an annotation to an object, run the following command:

```
cf curl v3/OBJECT-ENDPOINT/GUID 
-X PATCH
-d '{
   "metadata": {
      "annotations": {
         "ANNOTATION-KEY": "ANNOTATION-VALUE"
      }
   }
}'
```

Where:

• **OBJECT-ENDPOINT** is the CAPI endpoint for the type of object you want to label, such as `apps` or `organizations`

• **GUID** is the GUID of the object you want to label

• **ANNOTATION-KEY** is the key for the annotation key

• **ANNOTATION-VALUE** is the corresponding value for the annotation key

For example, the following command provides a `contacts` annotation for an app.

```
$ cf curl v3/apps/1cb006ee-fb05-47e1-b541-c34179ddc446 
-X PATCH
-d '{
   "metadata": {
      "annotations": {
         "contacts": "Bill tel(1111111) email(bill@fixme), Bob tel(2222222) pager(3333333)@fixme)"
      }
   }
}'
```

Update Metadata for an Object

To update metadata for an object, follow the procedure for adding metadata and provide a new value for an existing key. See Add Metadata to an Object above.

Remove Metadata from an Object

To remove metadata from an object, follow the procedure for adding metadata and provide a `null` value for an existing key. See Add Metadata to an Object above.

View Metadata for an Object

The following procedure describes how to view metadata:
1. To view metadata using the list endpoint of an object, run the following command:

```
cf curl /v3/OBJECT-ENDPOINT/GUID
```

Where:

- **OBJECT-ENDPOINT** is the CAPI endpoint for the type of object you want to view, such as `apps` or `organizations`
- **GUID** is the GUID of the object you want to view

For example:

```
$ cf curl /v3/apps/1cb006ee-fb05-47e1-b541-c34179ddc446
{
  "guid": "1cb006ee-fb05-47e1-b541-c34179ddc446",
  "name": "my_app",
  "state": "STOPPED",
  "created_at": "2016-03-17T21:41:30Z",
  "updated_at": "2016-06-08T16:41:26Z",
  "lifecycle": {
    "type": "buildpack",
    "data": {
      "buildpacks": ["java_buildpack"],
      "stack": "cflinuxfs3"
    }
  },
  "relationships": {
    "space": {
      "data": {
        "guid": "2f35885d-0c9d-4423-83ad-fd05066f8576"
      }
    }
  },
  "links": {
    ...
  },
  "metadata": {
    "labels": {
      "environment": "production"
    },
    "annotations": {
      "contacts": "Bill tel(1111111) email(bill@fixme), Bob tel(2222222) pager(3333333#555) email(bob@fixme)"
    }
  }
}
```

List Objects by Querying Labels

The following procedure describes how to list objects by querying label metadata:

1. To query an object by using the `label_selector` parameter on its list endpoint, run the following command:

```
cf curl /v3/OBJECT-ENDPOINT/?label_selector=SELECTOR-REQUIREMENTS
```

Where:

- **OBJECT-ENDPOINT** is the CAPI endpoint for the type of object you want to view, such as `apps` or `organizations`
- **SELECTOR-REQUIREMENTS** is one of requirement types specified in `Selector Requirement Reference` below. You can add multiple selector requirements using a comma-separated list.

**Note:** Ensure that this part of the URL is appropriately escaped.

For example, the following command selects all apps tagged with a label that has a key `environment` and value `production`.

```
$ cf curl /v3/apps/?label_selector=environment:production
```

Selector Requirement Reference
The following table describes how to form selector requirements:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existence</td>
<td>KEY</td>
<td>Returns all resources labeled with the given key</td>
</tr>
<tr>
<td>existence</td>
<td>!KEY</td>
<td>Returns all resources not labeled with the given key</td>
</tr>
<tr>
<td>equality</td>
<td>KEY==VALUE</td>
<td>or KEY=VALUE</td>
</tr>
<tr>
<td>inequality</td>
<td>KEY!=VALUE</td>
<td>Returns all resources not labeled with the given key and value</td>
</tr>
<tr>
<td>set inclusion</td>
<td>KEY in (VALUE1,VALUE2...)</td>
<td>Returns all resources labeled with the given key and one of the specified values</td>
</tr>
<tr>
<td>set inclusion</td>
<td>KEY notin (VALUE1,VALUE2...)</td>
<td>Returns all resources not labeled with the given key and one of the specified values</td>
</tr>
</tbody>
</table>

Example: Label Objects with a Git Commit

This section provides the following:

- A procedure for labeling an app, package, and droplet with a git commit SHA. See Manually Label Objects.
- A script that automates the procedure. See Automate Labeling Objects.

Labeling your app and related resources with a git commit SHA allows you to track which version of your code is running on PAS.

For more information about app packages and droplets, see the CAPI documentation.

Manually Label Objects

To label an app, droplet, and package with a git commit SHA, do the following:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI).
2. Run the following command and record the app GUID:

   ```bash
cf app APP_NAME --guid
   ```

   Where `APP_NAME` is the name of the app.

3. Run the following command to return the GUID of the droplet and package associated with the app:

   ```bash
cf curl /v3/apps/APP_GUID/droplets/current
   ```

   Where `APP_GUID` is the GUID of the app.

4. Record the GUID of the droplet and package:
   - The droplet GUID is the value for the "guid" key.
   - The package GUID is the end of the "href" URL for the "package" key.

   For example, the droplet and package GUIDs are highlighted in blue in the following output:

   ```json
   {
   "guid": "fd35633f-5c5c-4e4e-a5a9-0722c970a9d2",
   "links": { 
   "package": {  
   "href": "https://api.run.pivotal.io/v3/packages/fd35633f-5c5c-4e4e-a5a9-0722c970a9d2"
   }
   }
   }
   ```

5. Run the following command to label the app with a git commit SHA:

   ```bash
cf curl /v3/apps/APP_GUID-X PATCH -f '{"metadata": { "labels": { "commit": COMMIT_SHA } } }'
   ```

   Where:
   - `APP_GUID` is the GUID of the app.
6. Run the following command to label the app droplet with the same git commit SHA:

```bash
cf curl /v3/droplets/DROPLET_GUID -X PATCH -d '{"metadata": {"labels": {"commit": COMMIT_SHA } } }
```

Where:
- **DROPLET_GUID** is the GUID of the droplet.
- **COMMIT_SHA** is the SHA of the git commit.

7. Run the following command to label the app package with the same git commit SHA:

```bash
cf curl /v3/packages/PACKAGE_GUID -X PATCH -d '{"metadata": {"labels": {"commit": COMMIT_SHA } } }
```

Where:
- **PACKAGE_GUID** is the GUID of the package.
- **COMMIT_SHA** is the SHA of the git commit.

## Automate Labeling Objects

You can automate labeling objects by running a script either programatically or manually in the app repository.

### Prerequisite

To run the following example script, you must install [jq](https://stedolan.github.io/jq/). To download [jq](https://stedolan.github.io/jq/), see [jq](https://stedolan.github.io/jq/).

### Example Script

The following script retrieves the GUID of the app, droplet, and package. It then `curl` CAPI to label each resource with the current git commit SHA.

Replace **APP-NAME** with the name of your app.

```bash
#!/usr/bin/env bash

set -ex

APP_GUID=$(cf app APP-NAME --guid)
APP_URI="/v3/apps/$APP_GUID"

DROPLET_GUID=$(cf curl /v3/apps/$APP_GUID/droplets/current | jq -r .guid)
DROPLET_URI="/v3/droplets/$DROPLET_GUID"

PACKAGE_GUID=$(cf curl /v3/droplets/$DROPLET_GUID | jq -r .links.package.href | xargs basename)
PACKAGE_URI="/v3/packages/$PACKAGE_GUID"

COMMIT_SHA=$(git rev-parse --short HEAD)
REQUEST_BODY=$(jq -n -r --arg commit "$COMMIT_SHA" '{"metadata": {"labels": {"commit": $commit } } }')

cf curl "$APP_URI" -X PATCH -d "$REQUEST_BODY"
cf curl "$PACKAGE_URI" -X PATCH -d "$REQUEST_BODY"
cf curl "$DROPLET_URI" -X PATCH -d "$REQUEST_BODY"
```
Getting Started with the Notifications Service

This topic describes how to use the Notifications Service, including how to create a client, obtain a token, register notifications, create a custom template, and send notifications to your users.

Prerequisites

- Install Pivotal Application Service (PAS).
- You must have admin permissions on your Cloud Foundry instance. You also must configure Application Security Groups (ASGs).
- Install the Cloud Foundry Command Line Interface (cf CLI) and User Account and Authorization Server (UAAC) command line tools.

Create a Client and Get a Token

To interact with the Notifications Service, you must create UAA scopes:

1. Use `uaac target uaa.YOUR-DOMAIN` to target your UAA server.

   ```
   $ uaac target uaa.example.com
   ```

2. Record the Admin Client Credentials from the UAA row in the PAS Credentials tab.

3. Use `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. UAAC stores the token in `~/.uaac.yml`.

   ```
   $ uaac token client get admin -s MyAdminPassword
   ```

4. Create a notifications-admin client with the required scopes.

   ```
   $ uaac client add notifications-admin --authorized_grant_types client_credentials --authorities notifications.write,notifications.manage,notifications.write,notification_templates.write,notification_templates.read,notification_templates.write!
   ```

   - `notifications.write`: send a notification. For example, you can send notifications to a user, space, or everyone in the system.
   - `notifications.manage`: update notifications and assign templates for that notification.
   - (Optional) `notification_templates.write`: create a custom template for a notification.
   - (Optional) `notification_templates.read`: check which templates are saved in the database.

5. Log in using your newly created client:

   ```
   $ uaac token client get notifications-admin
   ```

   Note: Stay logged in to this client to follow the examples in this topic.

Register Notifications

Note: To register notifications, you must have the `notifications.manage` scope on the client. To set critical notifications, you must have the `critical_notifications.write` scope.

You must register a notification before sending it. Using the token `notifications-admin` from the previous step, the following example registers two notifications with the following properties:
source_name has “Cloud Ops Team” set as the description.

system-going-down and system-up are the notifications set.

system-going-down and system-up are made critical, so no users can unsubscribe from that notification.

Create a Custom Template

A template is made up of a name, a subject, a text representation of the template you are sending for mail clients that do not support HTML, and an HTML version of the template.

The system provides a default template for all notifications, but you can create a custom template using the following curl command.

```
curl https://notifications.user.example.com/templates -X POST --data '{
"name": "site-maintenance",
"subject": "Maintenance: {{.Subject}}",
"text": "The site has gone down for maintenance. More information to follow {{.Text}}",
"html": "The site has gone down for maintenance. More information to follow {{.HTML}}"
}'
```

Variables that take the form `{{.}}` interpolate data provided in the send step before a notification is sent. Data that you can insert into a template during the send step include `{{.Text}}`, `{{.HTML}}`, and `{{.Subject}}`.

This curl command returns a unique template ID that can be used in subsequent calls to refer to your custom template. The result looks similar to this:

```
{"template-id": "E3710280-954B-4147-B7E2-AF5BF62772B5"}
```

Check all of your saved templates by running a curl command:

```
curl https://notifications.user.example.com/templates -X GET
```

Associate a Custom Template with a Notification

In this example, the system-going-down notification belonging to the notifications-admin client is associated with the template ID E3710280-954B-4147-B7E2-AF5BF62772B5. This is the template ID of the template we created in the previous section.

Associating a template with a notification requires the notifications.manage scope.

```
curl https://notifications.user.example.com/clients/notifications-admin/notifications/system-going-down/template -X PUT --data '{"template": "E3710280-954B-4147-B7E2-AF5BF62772B5"}'
```

Any notification that does not have a custom template applied, such as system-up, defaults to a system-provided template.

Send a Notification

You can send a notification to the following recipients:

- A user
- A space
- An organization
- All users in the system
- A UAA-scope
- An email address

For details, see the Notifications V1 Documentation in GitHub.

The following example sends the `system-going-down` notification described above to all users in the system.

```bash
5 user curl https://notifications.user.example.com/everyone -X POST --data \
'"{"kind_id":"system-going-down","text":"The system is going down while we upgrade our storage","html":""\nTHE SYSTEM IS DOWN\n"","subject":"Upgrade to Storage","reply_to":"no-reply@example.com"}'
```
Configuring Container-to-Container Networking

This topic describes how to configure the Container-to-Container Networking feature, which allows direct network traffic between apps. For an overview of how Container-to-Container Networking works, see the Container-to-Container Networking topic.

Container-to-Container Networking enables PCF to generate logs whenever containers communicate or attempt to communicate with each other. See App Traffic Logging for how to manage app traffic logging.

Configure the Overlay Network

Container-to-Container Networking uses an overlay network to manage communication between app instances. By default, each Diego cell in the overlay network is allocated a /24 range that supports 254 containers per cell, one container for each of the usable IP addresses, 1 through 254. For more information about the overlay network, see Overlay Network in Container-to-Container Networking.

Configure the Number of Diego Cells

If you want to modify the number of Diego cells supported by the overlay network, follow the steps below:

1. In Ops Manager, select the PAS tile.
2. Select Networking.
3. Under Overlay Subnet, enter an IP range for the overlay network. By default, Ops Manager uses 10.255.0.0/16. Modifying the subnet range allocated to the overlay network changes the number of Diego cells supported in your deployment. Use the table below as a reference.

<table>
<thead>
<tr>
<th>Overlay subnet mask</th>
<th>Number of cells</th>
<th>Containers per cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>/20</td>
<td>63</td>
<td>254</td>
</tr>
<tr>
<td>/16</td>
<td>255</td>
<td>254</td>
</tr>
<tr>
<td>/12</td>
<td>4,095</td>
<td>254</td>
</tr>
</tbody>
</table>

⚠️ warning: The overlay network IP address range must not conflict with any other IP addresses in the network. If a conflict exists, Diego cells cannot reach any endpoint that has a conflicting IP address.

Create and Manage Networking Policies

This section describes how to create and modify Container-to-Container Networking policies using the Cloud Foundry Command Line Interface (cf CLI). The cf CLI only supports configuring policies for apps within the same space. To configure policies for apps in different orgs and spaces, use the Policy Server External API.

⚠️ Note: You can also create and modify container-to-container networking policies using Apps Manager. For more information, see Create Container-to-Container Networking Policies.

⚠️ Note: With the NSX-T integration, container networking policies and ASGs continue to work as normal. Advanced ASG logging is not supported with NSX-T.

Prerequisites

- Ensure that you are using cf CLI v6.30 or higher:
  ```
  $ cf version
  ```
  For more information about updating the cf CLI, see the Installing the cf CLI topic.
Grant Permissions

CF admins use the following UAA scopes to grant specific users or groups permissions to configure network policies:

<table>
<thead>
<tr>
<th>UAA Scope</th>
<th>Suitable for...</th>
<th>Allows users to create policies...</th>
</tr>
</thead>
<tbody>
<tr>
<td>network.admin</td>
<td>operators</td>
<td>for any apps in the CF deployment</td>
</tr>
<tr>
<td>network.write</td>
<td>space developers</td>
<td>for apps in spaces that they can access</td>
</tr>
</tbody>
</table>

If you are a CF admin, you already have the network.admin scope. An admin can also grant the network.admin scope to a space developer.

For more information, see Creating and Managing Users with the UAA CLI (UAAC) and Orgs, Spaces, Roles, and Permissions.

To grant all Space Developers permissions to configure network policies, open the Application Developer Controls pane in your PAS tile and enable the Allow Space Developers to manage network policies checkbox.

Add a Network Policy

To add a policy that allows direct network traffic from one app to another, run the following command:

```bash
cf add-network-policy SOURCE_APP --destination-app DESTINATION_APP --protocol (tcp | udp) --port RANGE
```

Replace the placeholders in the above command as follows:

- **SOURCE_APP** is the name of the app that sends traffic.
- **DESTINATION_APP** is the name of the app that will receive traffic.
- **PROTOCOL** is one of the following: tcp or udp.
- **RANGE** are the ports at which to connect to the destination app. The allowed range is from 1 to 65535. You can specify a single port, such as 8080, or a range of ports, such as 8080-8090.

The following example command allows access from the *frontend* app to the *backend* app over TCP at port 8080:

```bash
$ cf add-network-policy frontend --destination-app backend --protocol tcp --port 8080
```

List Policies

You can list all the policies in your space, or just the policies for which a single app is the source:

- **To list the all the policies in your space**, run ```cf network-policies```.

```bash
$ cf network-policies
```

- **To list the policies for an app**, run ```cf network-policies --source MY-APP```. Replace ```MY-APP``` with the name of your app.

```bash
$ cf network-policies --source example-app
```

The following example command lists policies for the app *frontend*:

```bash
$ cf network-policies --source frontend
```

Remove a Network Policy

To remove a policy that allows direct network traffic from an app, run the following command:

```bash
$ cf remove-network-policy SOURCE_APP --destination-app DESTINATION_APP --protocol (tcp | udp) --port RANGE
```
Replace the placeholders in the above command to match an existing policy, as follows:

- **SOURCE_APP** is the name of the app that sends traffic.
- **DESTINATION_APP** is the name of the app that receives traffic.
- **PROTOCOL** is either `tcp` or `udp`.
- **PORTS** are the ports connecting the apps. The allowed range is from `1` to `65535`. You can specify a single port, such as `8080`, or a range of ports, such as `8080-8090`.

The following command deletes the policy that allowed the `frontend` app to communicate with the `backend` app over TCP on port 8080:

```
$ cf remove-network-policy frontend --destination-app backend --protocol tcp --port 8080
Removing network policy to app frontend in org my-org / space dev as admin...
OK
```

**Disable Network Policy Enforcement**

You can disable Silk network policy enforcement between apps. Disabling network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

To disable network policy enforcement between apps, do the following:

1. Go to the **Networking** pane of your PAS tile.
2. Disable the **Enable Silk Network Policy Enforcement** checkbox.
3. Click **Save**.

**App Service Discovery**

With app service discovery, apps pushed to Pivotal Application Service (PAS) can establish container-to-container communications through a known route served by internal BOSH DNS. This allows front end apps to easily connect with back end apps.

Note: The internal domain used for service discovery is `apps.internal` by default. Operators can modify this value in **Application Developer Controls** pane of the PAS tile.

To establish container-to-container communications between a front end and back end app, a developer:

1. Launches a back end app that publishes a local endpoint.
2. Maps a named route to the endpoint.
3. Creates a network policy that allows direct traffic from the front end to the back end app.
4. Launches the front end app.

See [Cats and Dogs with Service Discovery](https://github.com) in GitHub for an example, written in Go, that demonstrates communication between front end and back end apps.
Administering Dynamic Egress Policies (Beta)


Note: This is a beta feature.

This topic describes how to administer dynamic egress policies for the apps in your deployment.

Overview
You can create dynamic egress policies to allow your apps to communicate with external services, such as a MySQL database. The workflow is as follows:
Create a destination object with details about the external service that your app or space need access to, including an IP range.
Create an egress policy from the app or space to the destination object.
Dynamic egress policies provide the following benefits over Application Security Groups (ASGs):
You do not have to restart your apps when applying these policies, so there is no downtime.
The policies include in additional level of granularity: you can apply them to specific apps.

Enable the Dynamic Egress Feature
The dynamic egress feature is disabled by default. To enable this feature, you must select Enable Beta Dynamic Egress Enforcement in the PAS
Networking pane. Additionally, you must have Silk selected for your Container Network Interface Plugin.

API Authorization
To administer dynamic egress policies, you must have the network.admin UAA scope. If you are a CF admin, you already have the network.admin scope. An
admin can also grant the network.admin scope to a space developer. For more information, see Creating and Managing Users with the UAA CLI (UAAC) 
and Orgs, Spaces, Roles, and Permissions .
If you have the network.admin scope, you can interact with the API in the following ways:

Option 1: cf curl
When you use cf curl
following example:

, you do not have to provide a token or the full API URL because you are already logged in and targeting your deployment. See the

$ cf curl /networking/v1/external/egress_policies
{"total_egress_policies":2,"egress_policies":[{"source":{...}]}

Option 2: curl
When using curl , you must provide the token Authorization header and use the full API URL. See the following example:
$ export TOKEN=`cf oauth-token` # as CF admin
{"total_egress_policies":2,"egress_policies":[{"source":{...}]}

Typical Workflows
Dynamic egress policies depend on a destination object. See the following workflow examples:

© Copyright Pivotal Software Inc, 2013-2019

909

2.5


Add an Egress Policy

To create a policy, you must create a destination first:

1. Create an egress destination. See Create Egress Destinations.

2. Create an egress policy linking the destination and an app or space. Create Egress Policies.

3. See policy apply. No app restarts needed.

Update an Egress Policy when an IP Changes

To update a policy, you update the destination:

1. List all egress destinations to find the one you want to change. See List Egress Destinations.

2. Update the egress destination. See Update Egress Destinations.

3. See updated policy apply. No app restarts needed.

Egress Destination API

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/networking/v1/external/destinations</td>
<td>List Destinations</td>
</tr>
<tr>
<td>POST</td>
<td>/networking/v1/external/destinations</td>
<td>Create Destinations</td>
</tr>
<tr>
<td>PUT</td>
<td>/networking/v1/external/destinations</td>
<td>Update Destinations</td>
</tr>
<tr>
<td>DELETE</td>
<td>/networking/v1/external/destinations/GUID</td>
<td>Delete Destinations</td>
</tr>
</tbody>
</table>

List Egress Destinations

GET /networking/v1/external/destinations

Arguments:
- (optional) id: comma-separated id values. This cannot be used with name.
- (optional) name: comma-separated name values. This cannot be used with id.

Response Body:

This endpoint returns all egress destinations
Create Egress Destinations

**POST /networking/v1/external/destinations**

**Request Body:**

```
{
  "destinations": {
    {  
      "name": "oracle database",
      "description": "db for user accounts",
      "ips": [{"start": "1.9.9.9", "end": "1.9.9.20"},
               {"start": "8000", "end": "9000"}],
      "ports": [{"start": "8000", "end": "9000"}],
      "protocol": "tcp"
    },
    {  
      "name": "AWS",
      "ips": [{"start": "1.8.8.8", "end": "1.8.8.8"},
               {"start": "8000", "end": "9000"}],
      "ports": [{"start": "8000", "end": "9000"}],
      "protocol": "udp"
    }
  }
}
```

**Parameters**

<table>
<thead>
<tr>
<th>Field</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destinations.name</td>
<td>Y</td>
<td>The name of the destination. Must be globally unique.</td>
</tr>
<tr>
<td>destinations.description</td>
<td>N</td>
<td>A description of the destination.</td>
</tr>
<tr>
<td>destinations.ips.start*</td>
<td>Y</td>
<td>The start of the destination ip range. Must be IPv4.</td>
</tr>
<tr>
<td>destinations.ips.end*</td>
<td>Y</td>
<td>The end of the destination ip range. Must be IPv4. May be equal to the the start ip.</td>
</tr>
<tr>
<td>destinations.ports.start*</td>
<td>Y</td>
<td>The destination start port (1 - 65535). Ports are not applicable for ICMP protocol.</td>
</tr>
<tr>
<td>destinations.ports.end*</td>
<td>Y</td>
<td>The destination end port (1 - 65535). Ports are not applicable for ICMP protocol.</td>
</tr>
<tr>
<td>destinations.protocol</td>
<td>Y</td>
<td>The protocol (tcp, udp, or icmp)</td>
</tr>
<tr>
<td>destinations.icmp_type</td>
<td>N</td>
<td>The icmp type to allow when using the icmp protocol. Default is all icmp types, represented by -1.</td>
</tr>
<tr>
<td>destinations.icmp_code</td>
<td>N</td>
<td>The icmp code to allow when using the icmp protocol. Default is all icmp codes, represented by -1.</td>
</tr>
</tbody>
</table>

*Currently only one IP range and one port range is supported.
Update Egress Destinations

PUT /networking/v1/external/destinations

Request Body:

```
{
  "destinations": [
    {
      "id": "90be9c1f-b694-4463-9f1f-6ce71904440d",
      "name": "oracle database",
      "description": "db for user accounts",
      "ips": [{"start": "1.9.9.9", "end": "1.9.9.20"}],
      "ports": [{"start": 8000, "end": 9000}],
      "protocol": "tcp"
    },
    {
      "id": "7281341b-bd38-49e0-ace0-7bf5b7c54b87",
      "name": "AWS",
      "ips": [{"start": "1.8.8.8", "end": "1.8.8.8"}],
      "ports": [{"start": 8000, "end": 9000}],
      "protocol": "udp"
    }
  ]
}
```

Parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destinations.id</td>
<td>Y</td>
<td>The id of the destination. This id is returned in the destinations create response, as well as in the destinations index response.</td>
</tr>
<tr>
<td>destinations.name</td>
<td>Y</td>
<td>The name of the destination. Must be globally unique.</td>
</tr>
<tr>
<td>destinations.description</td>
<td>N</td>
<td>A description of the destination.</td>
</tr>
<tr>
<td>destinations.ips.start*</td>
<td>Y</td>
<td>The start of the destination ip range. Must be IPv4.</td>
</tr>
<tr>
<td>destinations.ips.end*</td>
<td>Y</td>
<td>The end of the destination ip range. Must be IPv4. May be equal to the the start ip.</td>
</tr>
<tr>
<td>destinations.ports.start*</td>
<td>Y</td>
<td>The destination start port (1 - 65535). Ports are not applicable for ICMP protocol.</td>
</tr>
<tr>
<td>destinations.ports.end*</td>
<td>Y</td>
<td>The destination end port (1 - 65535). Ports are not applicable for ICMP protocol.</td>
</tr>
<tr>
<td>destinations.protocol</td>
<td>Y</td>
<td>The protocol (tcp, udp, or icmp)</td>
</tr>
<tr>
<td>destinations.icmp_type</td>
<td>N</td>
<td>The icmp type to allow when using the icmp protocol. Default is all icmp types, represented by -1.</td>
</tr>
<tr>
<td>destinations.icmp_code</td>
<td>N</td>
<td>The icmp code to allow when using the icmp protocol. Default is all icmp codes, represented by -1.</td>
</tr>
</tbody>
</table>

*Note: Currently only one IP range and one port range is supported.

Delete an Egress Destination

DELETE /networking/v1/external/destinations/GUID

Response Body:

This endpoint returns the JSON of the deleted destination object.
Egress Policy API

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/networking/v1/external/egress_policies</td>
<td>List Egress Policies</td>
</tr>
<tr>
<td>POST</td>
<td>/networking/v1/external/egress_policies</td>
<td>Create Egress Policies</td>
</tr>
<tr>
<td>DELETE</td>
<td>/networking/v1/external/egress_policies/GUID</td>
<td>Delete Egress Policy</td>
</tr>
</tbody>
</table>

List Egress Policies

GET /networking/v1/external/egress_policies

Arguments:
None

Response Body:
This endpoint returns all egress policies.

Create Egress Policies

POST /networking/v1/external/egress_policies
Request Body:

```json
{
  "egress_policies": [{
    "source": {
      "type": "space",
      "id": "SOURCE-SPACE-GUID"
    },
    "destination": {
      "id": "EGRESS-DESTINATION-GUID"
    }
  }]
}
```

Parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source.type</td>
<td>N</td>
<td>The type of source. Must be ‘app’ or ‘space’. Defaults to ‘app’.</td>
</tr>
<tr>
<td>source.id</td>
<td>Y</td>
<td>The guid of the source app or space.</td>
</tr>
<tr>
<td>destination.id</td>
<td>Y</td>
<td>The guid of the egress destination.</td>
</tr>
</tbody>
</table>

Delete an Egress Policy

DELETE /networking/v1/external/egress_policies/GUID

Response Body:

This endpoint returns the JSON of the deleted egress policy object.

```json
{
  "total_egress_policies": 1,
  "egress_policies": [{
    "id": "dynamic-egress-guid",
    "source": {
      "type": "app",
      "id": "SOURCE-APP-GUID"
    },
    "destination": {
      "name": "AWS",
      "description": "AWS",
      "id": "72813418-49b6-38-4a0-acc0-7b567c54687",
      "ips": [{"start": "1.8.8.8", "end": "1.8.8.8"}],
      "ports": [{"start": 8000, "end": 9000}]
    }
  }
}
```
Managing Isolation Segments

This topic describes how operators can isolate deployment workloads into dedicated resource pools called isolation segments.

Requirements

You must have the v.6.26.0 version or later of the Cloud Foundry Command Line Interface (cf CLI) installed to manage isolation segments.

Target the API endpoint of your deployment with `cf api` and log in with `cf login` before performing the procedures in this topic. For more information, see the Identifying the API Endpoint for your PAS Instance topic.

Overview

To enable isolation segments, an operator must install the PCF Isolation Segment tile by performing the procedures in the Installing PCF Isolation Segment topic. Installing the tile creates a single isolation segment.

After an admin creates a new isolation segment, the admin can then create and manage relationships between the orgs and spaces of a Cloud Foundry deployment and the new isolation segment.

Create an Isolation Segment

Before you create an isolation segment in PCF, you must install the PCF Isolation Segment tile by performing the procedures in the Installing PCF Isolation Segment topic.

To register an isolation segment with Cloud Controller, use the cf CLI.

The following command creates an isolation segment named my_segment:

```
$ cf create-isolation-segment my_segment
```

If successful, the command returns an OK message:

```
Creating isolation segment my_segment as admin...
OK
```

Retrieve Isolation Segment Information

The `cf isolation-segments`, `cf org`, and `cf space` commands retrieve information about isolation segments. The isolation segments you can see depends on your role, as follows:

- **Admins** see all isolation segments in the system.
- **Other users** only see the isolation segments that their orgs are entitled to.

List Isolation Segments

The following request returns a list of the isolation segments that are available to you:

```
$ cf isolation-segments
```
For example, the command returns results similar to the following:

```
Getting isolation segments as admin...
OK
name       orgs
  my_segment   org1, org2
```

**Display Isolation Segments Enabled for an Org**

An admin can entitle an org to multiple isolation segments.

Run `cf org ORG-NAME` command to view the isolation segments that are available to an org. Replace `ORG-NAME` with the name of your org.

For example:

```
$ cf org my-org
```

The command returns results similar to the following:

```
Getting info for org my-org as user@example.com...
name:          my-org
domains:       example.com, apps.example.com
quota:         paid
spaces:        development, production, sample-apps, staging
isolation segments:  my_segment, my_other_segment
```

**Show the Isolation Segment Assigned to a Space**

Only one isolation segment can be assigned to a space.

Run `cf space SPACE-NAME` to view the isolation segment assigned to a space. Replace `SPACE-NAME` with the name of the space.

For example:

```
$ cf space staging
```

The command returns results similar to the following:

```
name:          staging
org:           my-org
apps:          
services:      isolation segment: my_segment
space quota:   security groups: dns, p-mysql, p-nginx, public_networks, rabbitmq, ssh-loggen
```

**Delete an Isolation Segment**

Note: An isolation segment with deployed apps cannot be deleted.

Only admins can delete isolation segments.

Run `cf delete-isolation-segment SEGMENT-NAME` to delete an isolation segment. Replace `SEGMENT-NAME` with the name of the isolation segment. If successful, the command returns an `OK` message.

For example:
Manage Isolation Segment Relationships

The commands listed in the sections below manage the relationships between isolation segments, orgs, and spaces.

Enable an Org to Use Isolation Segments

Only admins can enable orgs to use isolation segments. Run `cf enable-org-isolation ORG-NAME SEGMENT-NAME` to enable the use of an isolation segment. Replace `ORG-NAME` with the name of your org, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf enable-org-isolation org2 my_segment
```

If an org is entitled to use only one isolation segment, that isolation segment does not automatically become the default isolation segment for the org. You must explicitly set the default isolation segment of an org.

Disable an Org from Using Isolation Segments

Run `cf disable-org-isolation ORG-NAME SEGMENT-NAME` to disable an org from using an isolation segment. Replace `ORG-NAME` with the name of your org, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf disable-org-isolation org1 my_segment
```

If successful, the command returns an OK message:

```
Removing entitlement to isolation segment my_segment from org org1 as admin...
OK
```

Set the Default Isolation Segment for an Org

This section requires cf CLI v6.29.0 or later.

Only admins and org managers can set the default isolation segment for an org.

When an org has a default isolation segment, apps in its spaces belong to the default isolation segment unless you assign them to another isolation segment. You must restart running applications to move them into the default isolation segment.

Run `cf set-org-default-isolation-segment ORG-NAME SEGMENT-NAME` to set the default isolation segment for an org. Replace `ORG-NAME` with the name of your org, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf set-org-default-isolation-segment org1 my_segment
```

Setting isolation segment my_segment to default on org org1 as admin...
OK
To display the default isolation segment for an org, use the `cf org` command.

**Assign an Isolation Segment to a Space**

Admins and org managers can assign an isolation segment to a space. Apps in that space start in the specified isolation segment.

To assign an isolation segment to a space, you must first enable the space’s org to use the isolation segment. See [Enable an Org to Use Isolation Segments](#). Run `cf set-space-isolation-segment SPACE-NAME SEGMENT-NAME` to assign an isolation segment to a space. Replace `SPACE-NAME` with the name of the space, and `SEGMENT-NAME` with the name of the isolation segment.

For example:

```
$ cf set-space-isolation-segment space1 my_segment
```

**Reset the Isolation Segment Assignment for a Space**

Admins can reset the isolation segment assigned to a space to use the org’s default isolation segment.

Run `cf reset-space-isolation-segment SPACE-NAME` to assign the default isolation segment for an org to a space. Replace `SPACE-NAME` with the name of the space.

For example:

```
$ cf reset-space-isolation-segment space1
```
Routing for Isolation Segments

This topic describes how operators can configure and manage routing for isolation segments. Operators can deploy an additional set of routers for each isolation segment to handle requests for applications within the segment. This topic includes the following sections:

- Overview
- Step 1: Create Networks
- Step 2: Configure Networks for Routers
- Step 3: Configure Additional Routers
- Step 4: Add Routers to Load Balancers
- Step 5: Configure DNS and Load Balancers
- Step 6: Configure Firewall Rules

For more information about how isolation segments work, see the Isolation Segments section of the Cloud Foundry Security topic. For more information about creating isolation segments, see the Installing PCF Isolation Segment topic.

**Note:** The instructions in this topic assume you are using Google Cloud Platform (GCP). The procedures may differ on other IaaSes, but the concepts should be transferable.

### Overview

Isolation segments isolate the compute resources for one group of applications from another. However, these applications still share the same network resources. Requests for applications on all isolation segments, as well as for system components, transit the same load balancers and Cloud Foundry routers.

A shared Isolation Segment is the default isolation segment assigned to every org and space. This can be overwritten by assigning an explicit default for an organization. For more information about creating isolation segments, see the Installing PCF Isolation Segment topic.

The illustration below shows isolation segments sharing the same network resources.

Operators who want to prevent all isolation segments and system components from using the same network resources can deploy an additional set of routers for each isolation segment:
Use cases include:

- Requests for applications in an isolation segment must not share networking resources with requests for other applications.
- The Cloud Foundry management plane should only be accessible from a private network. As multiple IaaS load balancers cannot typically share the same pool of backends, such as Cloud Foundry routers, each load balancer requires an additional deployment of routers.

### Step 1: Create Networks

Create a network or subnet for each isolation segment on your infrastructure. For example, an operator who wants one shared isolation segment and one private segment could create one network named `sample-network` with two subnets named `sample-subnet-shared`, `sample-subnet-is1`.

The following diagram describes the network topology:

```
IaaS network: sample-network
  |   IaaS subnet: sample-subnet-shared
  |   IaaS subnet: sample-subnet-is1
Subnets do not generally span IaaS availability zones, so the same operator with two availability zones will need four subnets

IaaS network: sample-network
  |   IaaS subnet: sample-subnet-shared-az1
  |   IaaS subnet: sample-subnet-shared-az2
  |   IaaS subnet: sample-subnet-is1-az1
  |   IaaS subnet: sample-subnet-is1-az2
```

For more information about networks and subnets in GCP, see the Using Networks and Firewalls topic in the GCP documentation.

### Step 2: Configure Networks for Routers

To configure the subnets with Bosh, use Bosh Cloud Config subnets. Each subnet in the IaaS should correspond to a Bosh subnet that is labeled with the correct isolation segment.

Navigate to the Assign AZs and Networks section of the PCF Isolation Segment tile to assign your isolation segment to the network you created in Step 1. See the Installing PCF Isolation Segment topic for more information.
Step 3: Configure Additional Routers

Navigate to the Resource Config section of the PCF Isolation Segment tile and use the dropdown menu to set your Router instances to a number greater than zero. See the Installing PCF Isolation Segment topic for more information.

Step 4: Add Routers to Load Balancer

If your IaaS supports it, navigate to the Resource Config section of the PCF Isolation Segment tile and enter the name of your load balancer under Load Balancers. See the documentation specific to your IaaS in Installing Pivotal Cloud Foundry for more information. If your IaaS does not support this configuration, you must create static IP addresses and assign them to your load balancer out of band.

Step 5: Configure DNS and Load Balancers

Create a separate domain name for each router instance group, and configure DNS to resolve these domain names to a load balancer that routes requests to the matching routers.

Note: You must configure your load balancers to forward requests for a given domain to one router instance group only.

As router instance groups may be responsible for separate isolation segments, and an application may be deployed to only one isolation segment, requests should only reach a router that has access to the applications for that domain name. Load balancing requests for a domain across more than router instance group can result in request failures unless all the router instance groups have access to the isolation segments where applications for that domain are deployed.

Shared Domain Name

It is a common requirement for applications on separate isolation segments to be accessible at domain names that share a domain, such as private-domain.com. To achieve this configuration while also obeying the guideline for forwarding requests for a domain to only one router instance group, create a new Cloud Foundry domain for a needed subdomain, such as *.foo.private-domain.com.

The diagrams illustrate a topology with separate load balancers, but you could also use one load balancer with multiple interfaces. In this configuration:

- Requests for system domain *.cf-system.com and the shared domain *.shared-apps.com are forwarded to the routers for the shared isolation segment.
- Requests for private domain *.foo.private-domain.com are forwarded to the routers for IS1. Requests for private domain *.private-domain.com are forwarded to the routers for IS2.
Step 6: Configure Firewall Rules

Configure firewall rules to allow for necessary ingress and egress traffic for private and shared isolation segments. Assuming a default deny-all rule, properly configuring firewall rules prevents a request with a spoofed Host header from being forwarded by a router to an application in a different isolation segment.

To configure firewall rules for isolation segment traffic, do the following:

1. Configure the firewall rules in the table below:

```
<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Source</th>
<th>Allowed Protocols/Ports</th>
<th>Destination</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared-to-bosh</td>
<td>Shared isolation</td>
<td>tcp</td>
<td>BOSH Director</td>
<td>BOSH Agent on VMs in the shared isolation segment to reach BOSH Director</td>
</tr>
<tr>
<td>bosh-to-shared</td>
<td>BOSH Director</td>
<td>tcp</td>
<td>Shared isolation segment</td>
<td>BOSH director to control VMs in the shared isolation segment</td>
</tr>
<tr>
<td>shared-internal</td>
<td>Shared isolation</td>
<td>tcp</td>
<td>Shared isolation segment</td>
<td>VMs within the shared isolation segment to reach one another</td>
</tr>
<tr>
<td>shared-to-is1</td>
<td>Shared isolation</td>
<td>tcp:1801,8853</td>
<td>Private isolation segment</td>
<td>Diego BBS in shared isolation segment to reach Cells in private isolation segment</td>
</tr>
<tr>
<td>is1-to-bosh</td>
<td>Private isolation</td>
<td>tcp:4222,25250,25777</td>
<td>BOSH Director</td>
<td>BOSH agent on VMs in private isolation segment to reach BOSH Director</td>
</tr>
<tr>
<td>is1-internal</td>
<td>Private isolation</td>
<td>tcp</td>
<td>Private isolation segment</td>
<td>VMs within private isolation segment to reach one another</td>
</tr>
</tbody>
</table>
```

**Note:** Firewall rules are specific to each IaaS, so the exact definition of Source and Destination depends on the IaaS. For example, on GCP, a Source is a subnet and a Destination is a tag. On AWS, both Source and Destination are security groups.
2. (Optional) Configure the firewall rules in the table below:

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Source</th>
<th>Allowed Protocols/Ports</th>
<th>Destination</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>jumpbox-to-is1</td>
<td>Jumpbox VM</td>
<td>tcp:22</td>
<td>Private isolation segment</td>
<td>Jumpbox VMs to reach private isolation segment through SSH or BOSH SSH.</td>
</tr>
<tr>
<td>is1-to-jumpbox</td>
<td>Private isolation segment</td>
<td>tcp</td>
<td>Jumpbox VM</td>
<td>Private isolation segment to reach jumpbox VM. Opens reverse SSH or BOSH SSH tunnel from jumpbox VM to private isolation segment.</td>
</tr>
<tr>
<td>diego-cell-egress</td>
<td>Diego cell VM on private isolation segment</td>
<td>tcp</td>
<td>Internet</td>
<td>If Diego Cells must download buildpacks to stage applications, allow egress traffic from all Diego Cell VMs on private isolation segments to reach the Internet.</td>
</tr>
</tbody>
</table>

For more information about ports used by agents to communicate with BOSH, see [bosh-deployment](https://github.com) in GitHub.

For more information about networks and firewall rules for GCP, see [Using Subnetworks](https://cloud.google.com) in the GCP documentation.

**Port Reference Table**

See the following table to understand which protocols and ports map to which processes and manifest properties for the `is1-to-shared` rule above.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Process</th>
<th>Manifest Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>3000</td>
<td>Routing API</td>
<td>routing_api.port</td>
</tr>
<tr>
<td>tcp</td>
<td>3457</td>
<td>Doppler</td>
<td>metron_endpoint.dropsonde_port</td>
</tr>
<tr>
<td>tcp</td>
<td>4003</td>
<td>VXLAN Policy Agent</td>
<td>cf_networking.policy_server.internal_listen_port</td>
</tr>
<tr>
<td>tcp</td>
<td>4103</td>
<td>Silk Controller</td>
<td>cf_networking.silk_controller.listen_port</td>
</tr>
<tr>
<td>tcp</td>
<td>4222</td>
<td>NATS</td>
<td>router.nats.port</td>
</tr>
<tr>
<td>tcp</td>
<td>8080</td>
<td>Diego file server</td>
<td>diego.file_server.listen_addr</td>
</tr>
<tr>
<td>tcp</td>
<td>8082</td>
<td>Doppler gRPC</td>
<td>loggregator.doppler.grpc_port</td>
</tr>
<tr>
<td>tcp</td>
<td>8083</td>
<td>Reverse Log Proxy Gateway (cf auth proxy)</td>
<td>loggregator.reverse_log_proxy_gateway_cf_auth_proxy.proxy_port</td>
</tr>
<tr>
<td>tcp</td>
<td>8843</td>
<td>UAA</td>
<td>uaa.ssl.port</td>
</tr>
<tr>
<td>tcp</td>
<td>8844</td>
<td>CredHub</td>
<td>credhub.port</td>
</tr>
<tr>
<td>tcp</td>
<td>8853</td>
<td>BOSH DNS health</td>
<td>health.server.port from bosh-dns-release</td>
</tr>
<tr>
<td>tcp</td>
<td>8889</td>
<td>Diego BBS</td>
<td>diego.rep.bbs.api_location</td>
</tr>
<tr>
<td>tcp</td>
<td>8891</td>
<td>Diego Database (Locket)</td>
<td>diego.locket.listen_addr</td>
</tr>
<tr>
<td>tcp</td>
<td>9022</td>
<td>CC stager</td>
<td>capi.stager.cc.external_port</td>
</tr>
<tr>
<td>tcp</td>
<td>9023</td>
<td>CC TPS</td>
<td>capi.tps.cc.external_port</td>
</tr>
<tr>
<td>tcp</td>
<td>9090</td>
<td>CC uploader</td>
<td>http_port</td>
</tr>
</tbody>
</table>
Addtional GCP Information

For more information, see the following:

- “Backend Services” in the GCP documentation
- BOSH Google Compute Engine CPI Github repository

Sharding Routers for Isolation Segments

You can configure router sharding for isolation segments depending on your use case:

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
<th>How to Configure</th>
</tr>
</thead>
</table>
| Securing apps that run in an isolation segment| To provide security guarantees in addition to the firewall rules described above, you can configure sharding of the Gorouter’s routing table, resulting in a router dedicated for an isolation segment having knowledge only of routes for applications in the same isolation segment. | 1. In the Networking configuration pane of the Pivotal Application Service (PAS) tile, select the checkbox labeled Routers reject requests for Isolation Segments.  
2. Set the Router Sharding Mode in the isolation segment tile to Isolation Segment Only. |
| Deploying additional routers for PAS          | The flexibility of the configuration also supports deployment of a router that excludes all isolation segments. | 1. In the Networking configuration pane of the PAS tile, select the checkbox labeled Routers reject requests for Isolation Segments.  
2. Set the Router Sharding Mode in the isolation segment tile to No isolation Segment. |

**Note:** For compute isolation only, you can leave the Routers reject requests for isolation segments checkbox unselected in the PAS Networking pane. This is the default setting, which does not require any additional routers for the Isolation Segment tile.

Metrics for Routers Associated with Isolation Segments

For metrics emitted by the Gorouter, metrics can be distinguished by the name of the job. For example, the following line is a metric emitted of uptime:

```
origin:“gorouter” eventType:ValueMetric timestamp:1491338040750977602 deployment:“superman.cf-app.com” job:“router_is1” index:“9a4b639c-8f0b-4b2b-b332-4161ee4646e6” ip:“10.0.16.23” valueMetric:<name:“uptime” value:118 unit:“seconds” >
```
Using Feature Flags

Page last updated:

This topic describes how Cloud Foundry (CF) administrators can set feature flags using the Cloud Foundry Command Line Interface (cf CLI) to enable or disable the features available to users.

View and Edit Feature Flags

To perform the following procedures, you must be logged in to your deployment as an administrator using the cf CLI.

1. Use the `cf feature-flags` command to list the feature flags:

   ```
   $ cf feature-flags
   
   Features                  State
   user_org_creation         disabled
   private_domain_creation   enabled
   app_bin_upload            enabled
   app_scaling               enabled
   route_creation            enabled
   service_instance_creation enabled
   diego_docker              disabled
   set_roles_by_username     enabled
   unset_roles_by_username   enabled
   task_creation             enabled
   env_var_visibility        enabled
   spaceScoped_private_entry_creation enabled
   space_developer_env_var_visibility enabled
   service_instance_sharing  disabled
   hide_marketplace_from_unauthenticated_users disabled
   
   For descriptions of the features enabled by each feature flag, see the Feature Flags section below.
   
   2. To view the status of a feature flag, use the `cf feature-flag FEATURE-FLAG-NAME` command:

   ```
   $ cf feature-flag user_org_creation
   Retrieving status of user_org_creation as admin...
   OK
   Features         State
   user_org_creation disabled
   
   3. To enable a feature flag, use the `cf enable-feature-flag FEATURE-FLAG-NAME` command:

   ```
   $ cf enable-feature-flag user_org_creation
   
   4. To disable a feature flag, use the `cf disable-feature-flag FEATURE-FLAG-NAME` command:

   ```
   $ cf disable-feature-flag user_org_creation

Feature Flags

Only administrators can set feature flags. All flags are enabled by default except `user_org_creation` and `diego_docker`. When disabled, these features are only available to administrators.

The following list provides descriptions of the features enabled or disabled by each flag, and the minimum Cloud Controller API (CC API) version necessary to use the feature. To determine your CC API version, follow the instructions in Identifying the API Endpoint for your PAS Instance.

- **user_org_creation**: Any user can create an organization. If enabled, this flag activates the Create a New Org link in the dropdown menu of the left navigation menu in Apps Manager. Minimum CC API version: 2.12.
- **private_domain_creation**: An Org Manager can create private domains for that organization. Minimum CC API version: 2.12.
- **app_bits_upload**: Space Developers can upload app bits. Minimum CC API version: 2.12.
- **app_scaling**: Space Developers can perform scaling operations (i.e. change memory, disk, or instances). Minimum CC API version: 2.12.
- **route_creation**: Space Developers can create routes in a space. Minimum CC API version: 2.12.
- **service_instance_creation**: Space Developers can create service instances in a space. Minimum CC API version: 2.12.
- **diego_docker**: Space Developers can push Docker apps. Minimum CC API version 2.33.
- **set_roles_by_username**: Org Managers and Space Managers can add roles by username. Minimum CC API version: 2.37.
- **unset_roles_by_username**: Org Managers and Space Managers can remove roles by username. Minimum CC API version: 2.37.
- **task_creation**: Space Developers can create tasks on their application. Minimum CC API version: 2.47.
- **env_var_visibility**: All users can view environment variables. Minimum CC API version: 2.58.
- **spaceScoped_private_broker_creation**: Space Developers can create space-scoped private service brokers. Minimum CC API version: 2.58.
- **space_developer_env_var_visibility**: Space Developers can view their v2 environment variables. Org Managers and Space Managers can view their v3 environment variables. Minimum CC API version: 2.58.
- **service_instance_sharing**: Space Developers can share service instances between two spaces (across orgs) in which they have the Space Developer role.
- **hide_marketplace_from_unauthenticated_users**: Do not allow unauthenticated users to see the service offerings available in the marketplace.

For more information about feature flag commands, see the Feature Flags section of the Cloud Foundry API documentation.

### Disabling Custom Buildpacks

You can disable custom buildpacks using your Ops Manager PAS tile. From the Cloud Controller tab, check the Disable Custom Buildpacks checkbox, as shown in the image below.

By default, the cf CLI gives developers the option of using a custom buildpack when they deploy apps to PAS. To do so, they use the `cf push` command. The Disable Custom Buildpacks checkbox prevents the `cf push` option from being used with external buildpack URLs.

For more information about custom buildpacks, refer to the buildpacks section of the PCF documentation.
Stopping and Starting Virtual Machines

Page last updated:

This topic assumes you are using BOSH CLI version 2.

This topic describes stopping and starting the component virtual machines (VMs) that make up a Pivotal Cloud Foundry (PCF) deployment.

In some cases you may want to stop all your VMs (for example, power down your deployment) or start all of your PAS VMs (for example, recover from a power outage.) You can stop or start all PAS VMs with a single `bosh` command.

If you want to shut down or start up a single VM in your deployment, you can use the manual process described on this page.

This procedure uses the BOSH Command Line Interface (BOSH CLI). See Advanced Troubleshooting with the BOSH CLI for more information about using this tool.

Stopping and Starting All PAS VMs

This section describes how to stop and start all the VMs in your deployment.

Stopping All PAS VMs

To shut down all the VMs in your deployment, perform the following steps:

1. Scale down the following jobs to one instance:
   - consul_server
   - mysql

2. Run the following command for each of the deployments listed in the previous step:

   
   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT stop --hard
   
   Replace the text above with the following:
   ```
   ```
   MY-ENV : the alias you set for the BOSH Director.
   MY-DEPLOYMENT : the name of your deployment.
   ```
   For example:

   ```
   $ bosh -e prod -d mysql stop --hard
   ```
   
   This command stops all VMs in the specified deployment. The `--hard` flag instructs BOSH to delete the VMs but retain any persistent disks.

Starting All PAS VMs

Perform the following steps to start all the VMs in your deployment:

1. Select the product deployment for the VMs you want to shut down. You can run the following command to locate CF deployment manifests:

   ```
   $ find /var/tempest/workspaces/default/deployments -name cf-* .yml
   ```

2. Run the following command:

   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT start
   ```

   Replace the text above with the following:

   ```
   MY-ENV : the alias you set for the BOSH Director.
   MY-DEPLOYMENT : the name of your deployment.
   ```
For example:

```bash
$ bosh -e prod -d mysql start
```

This command starts all VMs in the specified deployment.

3. If you require high availability in your deployment, scale up all instance groups to the original or desired counts.

### Stopping and Starting Individual PAS VMs

This section describes how to stop and start individual VMs in your deployment.

#### Find the Names of Your PAS Virtual Machines

You need the full names of the VMs to stop and start them using the BOSH CLI. To find full names for the VMs running each component, run

```bash
bosh -e MY-ENV instances
```

replacing `MY-ENV` with the alias you set for your BOSH Director. To filter the list of instances by deployment, run

```bash
bosh -e MY-ENV -d MY-DEPLOYMENT instances
```

For example:

```bash
$ bosh -e prod -d mysql instances
```

![Instance output](image)

You can see the full name of each VM in the `Instance` column of the terminal output. Each full name includes:

- A prefix indicating the component function of the VM.
- An identifier string specific to the VM.

For any component, you can look for its prefix in the `bosh instances` output to find the full name of the VM or VMs that run it.

#### Stopping an Individual PAS VM

To stop a job, run the following command for the component in your PAS deployment, replacing `MY-ENV` with the alias you set for your BOSH Director and `MY-DEPLOYMENT` with the name of the deployment:

```bash
bosh -e MY-ENV -d MY-DEPLOYMENT stop VM-NAME
```

To delete the instance that contains the job, run the following command for the component in your PAS deployment:

```bash
bosh -e MY-ENV -d MY-DEPLOYMENT stop VM-NAME --hard
```

Use the full name of the component VM as listed in your `bosh instances` terminal output without the unique identifier string.

For example, the following command stops the Loggregator Traffic Controller job:

```bash
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller
```

To stop a specific instance of a job, include the identifier string at the end of its full name.

For example, the following command stops the Loggregator Traffic Controller job on only one Diego cell instance:
To delete the VM, include `--hard` at the end of the command. This command does not delete persistent disks.

For example, the following command deletes a specific Loggregator Traffic Controller instance:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller/0123-abcd-4567ef89 --hard
```

## Starting an Individual PAS VM

Run the following command for the component in your PAS deployment you wish to start, replacing `MY-ENV` with the alias you set for your BOSH Director and `MY-DEPLOYMENT` with the name of the deployment. Use the full name of the component VM as listed in your `bosh vms` terminal output without the unique identifier string.

```
bosh -e MY-ENV -d MY-DEPLOYMENT start VM-NAME
```

The following example command starts the Loggregator Traffic Controller VM:

```
$ bosh -e prod -d loggregator start loggregator_trafficcontroller
```

To start a specific instance of a VM, include the identifier string at the end of its full name.

For example, the following command starts the Loggregator Traffic Controller job on one Diego cell instance:

```
$ bosh -e prod -d loggregator start loggregator_trafficcontroller/0123-abcd-4567ef89
```
Examining GrootFS Disk Usage

This topic describes how to analyze GrootFS disk space usage in Pivotal Application Service (PAS).

You run the commands in this topic on any BOSH-deployed VM that hosts the Garden job in a PAS deployment.

**Note:** This topic provides different commands depending on whether you are using privileged or unprivileged containers in your deployment. By default, all deployments use unprivileged containers. For more information about these container types, see Container Security.

For more information about the GrootFS concepts, see GrootFS Disk Usage.

Reconcile Container Disk Usage and Host Disk Usage

To reconcile disk allocations for containers with the actual disk usage on the host VM, you need to understand how GrootFS uses its disk.

Command line tools such as `du` and `df` can provide misleading information because of the way container file systems work.

For example, the following situations can occur:

<table>
<thead>
<tr>
<th>If...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>the Diego cell rep appears to be out of disk capacity, but the actual disk usage on the Garden host is low</td>
<td>Diego does not schedule containers on the cell.</td>
</tr>
<tr>
<td>the Diego cell rep appears to have available disk capacity, but the combined space used by containers and system components prevents Diego from allocating the remaining disk space</td>
<td>Diego continues to place containers on the cell, but they fail to start.</td>
</tr>
</tbody>
</table>

About Container Disk Usage

On disk, the read-write layer for each container can be found at `/var/vcap/data/grootfs/store/unprivileged/images/CONTAINER-ID/diff`.

When GrootFS calls on the built-in XFS quota tooling to get disk usage for a container, it takes into account data written to that directory and not the data in the read-only volumes.

Running `grootfs stats` returns the following values:

- `total_bytes_used`: This is the disk usage of the container including the rootfs image volumes.
- `exclusive_bytes_used`: This is the disk usage of the container not including the rootfs image volumes.

Retrieve Disk Usage Stats for a Single Container

To obtain the disk usage stats of a single container, perform the following steps:

1. In your PAS deployment, use `bosh ssh` to connect to the BOSH-deployed VM running the Garden job.

2. On the VM, run the following command to look up the container ID:

   ```
   ls /var/vcap/data/garden/depot/
   ```

   The command above returns a container ID in the following format:

   ```
   55afbf65-5cbf-49c6-4461-f803
   ```

3. Based on the container type used in your deployment, run one of the following commands on the VM:

   - For unprivileged containers, run the following command:
Retrieve Exclusive Disk Usage Stats for All Running Containers

To check the disk usage of all running containers, perform the following steps:

1. In your PAS deployment, use `bosh ssh` to connect to the BOSH-deployed VM running the Garden job.

2. Based on the container type used in your deployment, run one of the following commands on the VM:

   - For unprivileged containers, run the following command:

     ```bash
     ls /var/vcap/data/garden/depot/ | xargs -I{} /var/vcap/packages/grootfs/bin/grootfs --config /var/vcap/jobs/garden/config/grootfs_config.yml stats {} | cut -d: -f4 | cut -d} -f1 | awk '{sum += $1}; END {print sum}'
     ```

   - For privileged containers, run the following command:

     ```bash
     ls /var/vcap/data/garden/depot/ | xargs -I{} /var/vcap/packages/grootfs/bin/grootfs --config /var/vcap/jobs/garden/config/privileged_grootfs_config.yml stats {} | cut -d: -f4 | cut -d} -f1 | awk '{sum += $1}; END {print sum}'
     ```

The commands above return the total disk usage in bytes for all running containers.

About Volumes in GrootFS

Underlying layers are known as volumes in GrootFS.

They are read-only and their changesets are layered together through an OverlayFS mount to create the rootfs for containers. When GrootFS writes each file system volume to disk, it also stores the number of bytes written to a file in the meta directory.

Check Volume Disk Size

To find out the size of an individual volume, you can read the corresponding metadata file or run `du` on the volume itself. Perform the following steps:

1. In your PAS deployment, use `bosh ssh` to connect to the BOSH-deployed VM running the Garden job.

2. Based on the container type used in your deployment, run one of the following commands on the VM:
For unprivileged containers, run the following command on the VM:
cat /var/vcap/data/grootfs/store/unprivileged/meta/volume-VOLUME-SHA

Where VOLUME-SHA corresponds to the SHA value of the volume.
For privileged containers, run the following command:
cat /var/vcap/data/grootfs/store/privileged/meta/volume-VOLUME-SHA

Where VOLUME-SHA corresponds to the SHA value of the volume.
The cat commands above return the volume size in bytes in the following format:
{"Size":5607885}

3. Alternatively, use du and pass the absolute path to the volume. Run one of the following commands on the VM:
For unprivileged containers, run the following command:
du -sch /var/vcap/data/grootfs/store/unprivileged/volumes/VOLUME-SHA/

Where VOLUME-SHA corresponds to the SHA value of the volume.
For privileged containers, run the following command:
du -sch /var/vcap/data/grootfs/store/privileged/volumes/VOLUME-SHA/

Where VOLUME-SHA corresponds to the SHA value of the volume.
The du commands above return the volume size in the following format:
5.4M

/var/vcap/data/grootfs/store/unprivileged/volumes/VOLUME-SHA/

Determine Disk Usage and Reclaimable Disk Space
This section describes how to calculate the amount of disk space is in use and estimate how much space is reclaimable.

Calculate Disk Use by All Active Volumes
For each container, GrootFS mounts the underlying volumes using overlay to a point in the images directory. This point is the rootfs for the container and
is read-write.
GrootFS also stores the SHA of each underlying volume used by an image in the meta folder.
You can determine the bytes of all active volumes on disk by running one of the following commands:
For unprivileged containers, run the following command:
for image in $(ls /var/vcap/data/grootfs/store/unprivileged/meta/dependencies/image\:*.json); \
do cat $image | python -c 'import json,sys;obj=json.load(sys.stdin); \
print "\n".join(obj)' ; done | sort -u \
| xargs -I{} cat /var/vcap/data/grootfs/store/unprivileged/meta/volume-{} \
| cut -d : -f 2 | cut -d} -f1 \
| awk '{sum += $1} END {print sum}'

For privileged containers, run the following command:
for image in $(ls /var/vcap/data/grootfs/store/privileged/meta/dependencies/image\:*.json); \
do cat $image | python -c 'import json,sys;obj=json.load(sys.stdin); \
print "\n".join(obj)' ; done | sort -u \
| xargs -I{} cat /var/vcap/data/grootfs/store/privileged/meta/volume-{} \
| cut -d : -f 2 | cut -d} -f1 \
| awk '{sum += $1} END {print sum}'

The commands above return the total number of bytes used by all active volumes on disk.

© Copyright Pivotal Software Inc, 2013-2019

932

2.5


Calculate GrootFS Store Disk Usage

To determine how much total disk space the store is using, run the following command:

```
df \ grep -E "/var/vcap/data/grootfs/store/(privileged|unprivileged)" \
| awk '{sum += $3} END {print sum}'
```

The command above returns the total number of bytes used by the store.

Calculate Reclaimable Disk Space

You can use values gathered from the commands above to calculate how much space can be cleared in GrootFS. Garbage collection reclaims disk space by pruning unused volumes.

The overall formula to calculate reclaimable disk space is to subtract the total disk in use by the store from the total disk used by active volumes.

For example, perform the following steps:

1. Calculate how much disk space the store is using by following the instructions in Calculate GrootFS Store Disk Usage. For example, your result might be:

   ```
   Total disk store = 5607885 bytes
   ```

2. Calculate how much disk space active volumes are using by following the instructions in Calculate Disk Use by All Active Volumes. For example, your result might be:

   ```
   Active volumes = 3212435 bytes
   ```

3. Subtract the amount of space used by active volumes from the space used by the store. For example, your result might be:

   ```
   5607885 - 3212435 = 2395450 bytes
   ```

In this example, you can reclaim 2395450 bytes through garbage collection.

How GrootFS Reclaims Disk Space

The thresholder component calculates and sets a value so that GrootFS’s garbage collector can attempt to ensure that a small reserved space is kept free for other jobs. GrootFS only tries to garbage collect or reclaim space when that threshold is reached. However, if all the rootfs layers are actively in use by images, then garbage collection cannot occur and that space is used up.

If you determine that there is not enough reclaimable disk and more space is needed on disk, you should scale up your VMs to a larger size or add more VMs to provide more disk space.

Alternatively, you can configure a lower threshold for cell disk cleanup in Pivotal Application Service (PAS). For more information, see Configuring Disk Cell Cleanup.

Other Categories of GrootFS Disk Usage

There may be categories of GrootFS disk usage other than those listed in the above sections. However, the bulk of disk usage is stored in the `images/CONTAINER-ID/diff` and `volumes` directories, so these are rarely taken into consideration when calculating store usage.

You can find these directories under `/var/vcap/data/grootfs/store/unprivileged` for unprivileged containers and `/var/vcap/data/grootfs/store/privileged` for privileged containers.

GrootFS also stores information in the following directories:

- `link`: link directories. Shorter directory names are symlinked to volume directories to allow Groot to union mount more file paths.
- `locka`: file system lock directory to ensure safety during concurrent cleans and creates.
- `meta`: per image and volume metadata.
- `projectIds`: empty numbered directories used to track image quotas.
- `tmp`: normal temporary directory contents.

These directories typically use less than 2 MB disk in total.
Enabling IPv6 for Hosted Applications

The procedure described below allows apps deployed to Pivotal Application Service (PAS) to be reached using IPv6 addresses.

**Note:** Amazon Web Services (AWS) EC2 instances currently do not support IPv6.

Pivotal Application Service (PAS) system components use a separate DNS subdomain from hosted applications. These components currently support only IPv4 DNS resolved addresses. This means that although an IPv6 address can be used for application domains, the system domain must resolve to an IPv4 address.

Complete the following steps to enable support for IPv6 application domains:

1. Set up an external load balancer for your Pivotal Application Service (PAS) deployment. See [Using Your Own Load Balancer](#).

2. Configure DNS to resolve application domains to an IPv6 address on your external load balancer. **Note:** Your IPv4 interface for the system domain and IPv6 interface for application domain can be configured on the same or different load balancers.

3. Configure the external load balancer to route requests for an IPv6 address to an IPv4 address as follows:
   - If you are using the HAProxy load balancer for SSL termination, route to its IPv4 address.
   - Otherwise, route directly to the IPv4 addresses of the Gorouters.

The following diagram illustrates how a single load balancer can support traffic on both IPv4 and IPv6 addresses for a Pivotal Application Service (PAS) installation.

![Diagram showing IPv4 and IPv6 traffic routing](image)

See [Routes and Domains](#) for more information about domains in Pivotal Application Service (PAS).
Securing Traffic into Cloud Foundry

This topic describes the options for securing HTTP traffic into your Pivotal Application Service (PAS) deployment with TLS certificates. You can configure the location where your deployment terminates TLS depending on your needs and certificate restrictions.

Protocol Support

The Gorouter supports HTTP/HTTPS requests only. For more information about features supported by the Gorouter, see the HTTP Routing topic.

You can force HTTPS-only connections by enabling HSTS on HAProxy. For more information, see Secure Apps Domain with HAProxy.

To secure non-HTTP traffic over TCP Routing, terminate TLS at your load balancer or at the application. See TCP Routing for details.

TLS Termination Options for HTTP Routing

There are several options for terminating TLS for HTTP traffic. You can terminate TLS at the Gorouter, your load balancer, or at both.

The following table summarizes TLS termination options and which option to choose for your deployment.

<table>
<thead>
<tr>
<th>If the following applies to you:</th>
<th>Then configure TLS termination at:</th>
<th>Related topic and configuration procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>You want the optimum balance of performance and security, and</td>
<td>Gorouter only</td>
<td>Terminating TLS at the Gorouter Only</td>
</tr>
<tr>
<td>You want to make minimum changes to your load balancer, or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You are deploying CF to AWS. For information about AWS limitations, see TLS Cipher Suite Support by AWS ELBs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You require TLS termination at a load balancer, or</td>
<td>Load Balancer and Gorouter</td>
<td>Terminating TLS at the Load Balancer and Gorouter</td>
</tr>
<tr>
<td>You want the highest level of security, and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You do not mind a slightly less performant deployment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You require TLS termination at a load balancer, and</td>
<td>Load Balancer only</td>
<td>Terminating TLS at the Load Balancer Only</td>
</tr>
<tr>
<td>You prefer unencrypted traffic between the Load Balancer and the Gorouter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optionally, if you are deploying HAProxy, and</td>
<td>Then in addition, terminate SSL/TLS at:</td>
<td>Related topic and configuration procedure</td>
</tr>
<tr>
<td>You would like to secure traffic to the HAProxy.</td>
<td>HAProxy</td>
<td>Terminating SSL/TLS at HAProxy</td>
</tr>
</tbody>
</table>

Certificate Requirements

The following requirements apply to the certificates you use to secure traffic into PAS:

- You must obtain at least one TLS certificate for your environment.
  - In a production environment, use a signed TLS certificate (trusted) from a known certificate authority (CA).
  - In a development or testing environment, you can use a trusted CA certificate or a self-signed certificate. You can generate a self-signed certificate with openssl or a similar tool.

Alternately, you can use the PAS Ops Manager interface to generate a certificate for you. Certificates generated in PAS are signed by the Ops Manager Certificate Authority. They are not technically self-signed, but they are sometimes referred to as “Self-Signed Certificates” in the Ops Manager UI and throughout this documentation.
Certificates used in CF must be encoded in the PEM format.

The Gorouter supports mutual TLS, and validates a client provided certificate chain against its CA certificates if one is provided in the TLS handshake, but does not require it. Depending on whether you choose to terminate at both the Load Balancer and the Gorouter, or at the Gorouter alone, the client certificate may be that of the load balancer or of the originating client.

The certificate on the Gorouter must be associated with the correct hostname so that HTTPS can validate the request.

If wildcard certificates are not supported for some or all of your domains, then configure termination requests at the load balancer only. In this type of deployment, the load balancer passes unencrypted traffic to the Gorouter. As a result, you avoid having to reissue and reinstall certificates on the Gorouter for every app or UAA security zone.

Extended Validation (EV) certificates support multiple hostnames, like SAN, but do not support wildcards. As the Gorouter has not been tested with EV certificates, if EV certificates are required, then terminate TLS at the load balancer only.

Given the dynamic and multi-tenant nature of PAS, use of wildcard domains is highly recommended to avoid the need for adding an additional certificate for each application.

Multiple Certificates

In order to support custom domains on CF, an operator has to configure the Gorouter with a certificate that represents the domain. It is recommended that operators add a new certificate instead of reissuing a single certificate when adding TLS support for an additional domain. Using multiple certificates provides a security benefit in that it prevents clients from discovering all the custom domains of applications running on a CF platform.

The Gorouter supports SNI and can be configured with multiple certificates, each which may optionally include wildcard and alternative names. The Gorouter uses SNI to determine the correct certificate to present in a TLS handshake. It requires clients to support the SNI protocol by sending a server name outside the encrypted request payload. For clients that do not support SNI, the Gorouter presents a default certificate. The default is the first certificate keypair in the Gorouter’s configuration.

The Gorouter decides which certificate to provide in the TLS handshake as follows:

- If a client provides an SNI header with a ServerName that matches to a configured certificate keypair, the Gorouter returns the matching certificate.
- If a client provides an SNI header with a ServerName that does not match a configured certificate keypair, the Gorouter returns the default certificate.

The first certificate keypair listed is used as the default.

The Gorouter supports both RSA and ECDSA certificates in PEM encoding. In the case that a certificate chain is required, the order should be as follows: primary certificate, intermediate certificate, then root certificate.

How to Configure Multiple Certificate Keypairs

To configure multiple HTTPS certificate keypairs for PAS, add each keypair along with a meaningful name in the applicable Certificates and Private Keys for HAProxy and Router fields of the Networking configuration screen in PAS. For more information, see Configuring PAS.

In PCF, multiple certificates configured for the Gorouter are also configured for HAPRoxy.

TLS Cipher Suite Support

Some CF components like the Gorouter support additional TLS cipher suites to accommodate older clients. As a security best practice, only configure the TLS cipher suites that you need for your deployment.

Default Gorouter Cipher Suites

By default, the Gorouter supports the following TLS cipher suites, both of which require TLS v1.2:

<table>
<thead>
<tr>
<th>RFC</th>
<th>OpenSSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>ECDHE-RSA-AES128-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>ECDHE-RSA-AES256-GCM-SHA384</td>
</tr>
</tbody>
</table>

You can override the default cipher suites in the TLS Cipher Suites for Router and Minimum version of TLS fields in the Networking tab of the PAS tile. See the following procedures for either Gorouter Only or Load Balancer and Gorouter for more information about using custom SSL ciphers.
TLS Cipher Suite Support by AWS Load Balancers

AWS Classic Load Balancers (formerly referred to as ELBs) support configuration of cipher suites for front-end connections with clients only. When configuring Classic Load Balancers to forward requests to Gorouters over TLS, operators may encounter a "Cipher Suite Mismatch" error. This is because the cipher suites supported by Classic Load Balancers for TLS handshakes with backends (Gorouters in this case) are hardcoded, undocumented, and do not support the Gorouter default cipher suites.

Operators have two options:

- Configure Classic Load Balancer listeners in TCP mode so that TCP connections from clients are passed through the Classic Load Balancer to Gorouters on port 443. Then Gorouters are the first point of TLS termination.
- If you require TLS termination at an AWS load balancer in addition to terminating at the Gorouter, use AWS Application Load Balancers (ALBs) that support the Gorouter default cipher suites.

TLS v1.2

The following cipher suites are optionally supported for TLS v1.2 only:

<table>
<thead>
<tr>
<th>RFC</th>
<th>OpenSSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_RSA_WITH_AES_128_GCM_SHA256</td>
<td>AES128-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_GCM_SHA384</td>
<td>AES256-GCM-SHA384</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_RC4_128_SHA</td>
<td>ECDHE-ECDSA-RC4-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</td>
<td>ECDHE-ECDSA-AES128-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</td>
<td>ECDHE-ECDSA-AES256-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_RC4_128_SHA</td>
<td>ECDHE-RSA-RC4-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>ECDHE-RSA-DES-CBC3-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</td>
<td>ECDHE-RSA-AES128-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</td>
<td>ECDHE-RSA-AES256-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>ECDHE-RSA-AES128-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</td>
<td>ECDHE-ECDSA-AES128-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256</td>
<td>ECDHE-RSA-AES256-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA256</td>
<td>AES128-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</td>
<td>ECDHE-ECDSA-AES128-GCM-SHA256</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA</td>
<td>AES128-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</td>
<td>ECDHE-RSA-AES256-SHA</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305</td>
<td>ECDHE-RSA-CHACHA20-POLY1305</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305</td>
<td>ECDHE-ECDSA-CHACHA20-POLY1305</td>
</tr>
</tbody>
</table>

TLS v1.0 and v1.1

The following cipher suites are optionally supported for TLS v1.0 and TLS v1.1 only:

<table>
<thead>
<tr>
<th>RFC</th>
<th>OpenSSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_RSA_WITH_RC4_128_SHA</td>
<td>RC4-SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>DES-CBC3-SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA</td>
<td>AES128-SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA</td>
<td>AES256-SHA</td>
</tr>
</tbody>
</table>

You can override the default cipher suites in the TLS Cipher Suites for Router and Minimum version of TLS fields in the Networking tab of the PAS tile. See the following procedures for either Gorouter Only or Load Balancer and Gorouter for more information about using custom SSL ciphers.

See Golang Constants and OpenSSL Cipher Suites for more information about supported ciphers.
**Mutual Authentication with Clients**

Gorouter supports validation of client certificates in TLS handshakes with clients, also known as mutual authentication. Operators can choose whether Gorouter requests client certificates and when requesting certificates, whether or not to require them.

By default, Gorouter requests but does not require client certificates in TLS handshakes.

To configure Gorouter behavior for handling client certificates, select one of the options in the **Router behavior for Client Certificates** field of the Networking configuration screen in PAS.

- **Router does not request client certificates.** The Gorouter does not request client certificates in TLS handshakes so clients will not provide them and validation of client certificates does not occur. This option is incompatible with the XFCC configuration options **TLS terminated for the first time at HAProxy** and **TLS terminated for the first time at the Router** in PAS because these options require mutual authentication.

- **Router requests but does not require client certificates.** The Gorouter requests client certificates in TLS handshakes. The handshake will fail if the client certificate is not signed by a CA configured for the router. This is the default configuration.

- **Router requires client certificates.** The Gorouter requests and requires client certificates in TLS handshakes. The handshake will fail if a client cert is not provided or if the client certificate is not signed by a CA configured for the router.

The behavior controlled by this property is global; it applies to all requests received by Gorouters so configured.

If Gorouter is the first point of TLS termination (your load balancer does not terminate TLS, and passes the request through to Gorouter over TCP), consider the following:

- Only option **Router does not request client certificates** should be used with PAS, as the Gorouters are in that product receive requests for the system domain. Many clients of CF platform APIs do not present client certificates in TLS handshakes, so the first point of TLS termination for requests to the system domain must not request them.

- All options may be used for routers deployed with the Isolation Segment tile, as these only receive requests for app domains.

- Options **Router requests but does not require client certificates** and **Router requires client certificates** will trigger browsers to prompt users to select a certificate if the browser is not already configured with a certificate signed by one of the CAs configured for the router.

If Gorouter is not the first point of TLS termination, this property can be used to secure communications between the Load Balancer and Gorouter. The router must be configured with the CA used to sign the client certification the load balancer will present.

⚠️ **warning:** Requests to the platform will fail upon upgrade if your load balancer is configured to present a client certificate in the TLS handshake with Gorouter but Gorouter has not been configured with the certificate authority used to sign it. To mitigate this issue, select **Router does not request client certificates** for **Router behavior for Client Certificate Validation** in the Networking pane or configure the router with the appropriate CA.

**Terminating TLS at the Gorouter Only**

In this configuration, the load balancer does not terminate TLS for CF domains at all. Instead, it passes through the underlying TCP connection to the Gorouter.

This option is the recommended and more performant option, establishing and terminating a single TLS connection.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.

Traffic between the load balancer and the Gorouter is encrypted only if the client request is encrypted.

---

© Copyright Pivotal Software Inc, 2013-2019 939 2.5

- **Note:** ECDSA ciphers require a certificate and key for DSA, as opposed to RSA.
Perform the following steps to configure SSL termination on the Gorouter in Pivotal Cloud Foundry (PCF):

1. Configure your load balancer to pass through TCP requests from the client to the Gorouter.

2. Navigate to the Ops Manager Installation Dashboard.

3. Click the Pivotal Application Service (PAS) tile in the Installation Dashboard.

4. Click Networking.

5. For PCF deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the Router IPs field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see Configuring PAS.

6. In the Certificates and Private Keys for HAProxy and Router field, click the Add button to define at least one certificate keypair for HAProxy and Router. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in PAS. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate for PCF Deployments.

7. In the Minimum version of TLS supported by HAProxy and Router field, select the minimum version of TLS to use in Gorouter communications. The Gorouter uses TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites.

8. Under HAProxy forwards requests to Router over TLS, select Disable.

9. If you want to use a specific set of TLS ciphers for the Gorouter, configure TLS Cipher Suites for Router. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites. Otherwise, leave the default values in this field.

10. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select Disable SSL certificate verification for this environment. Selecting this checkbox also disables SSL verification for route services.
11. (Optional) If you do not want the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on HAProxy and Router** checkbox.

12. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header field**, select the third option, **Strip the XFCC header when present and set it to the client certificate**.

13. Click **Save**.

14. In the **PAS tile**, click **Resource Config**.

15. In the **Instances** drop down for the **HAProxy** job, select 0 instances.

16. Click **Save**.

**Terminating TLS at the Load Balancer Only**

In this configuration, your load balancer terminates TLS, and passes unencrypted traffic to the Gorouter, which routes it to your app. Traffic between the load balancer and the Gorouter is not encrypted.

This option is recommended if you cannot use SAN certificates and if you do not require traffic to be encrypted between the load balancer and the Gorouter.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.

![TLS Termination at Load Balancer Diagram](image)

**About HTTP Header Forwarding**

If you terminate TLS at your load balancer, then you must also configure the load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` HTTP headers to the HTTP traffic it passes to the Gorouter.

For more information about HTTP headers in CF, see **HTTP Headers**. If you are configuring the forwarding of client certificates, see **Forward Client Certificate to Applications**.

**Procedure: Load Balancer Only**

Perform the following steps to configure SSL termination on the load balancer only in Pivotal Cloud Foundry (PCF):

1. Create an A record in your DNS that points to your load balancer IP address. The A record associates the System Domain and Apps Domain that you configure in the **Domains** section of the Pivotal Application Service (PAS) tile with the IP address of your load balancer.

   For example, with `cf.example.com` as the main subdomain for your Cloud Foundry deployment and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `198.51.100.1`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.cf</td>
<td>A</td>
<td>198.51.100.1</td>
<td>example.com</td>
</tr>
</tbody>
</table>

2. Navigate to the **Ops Manager Installation Dashboard**.

3. Click the **PAS tile** in the Installation Dashboard.

4. Click **Networking**.

5. For PCF deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them
in the **Router IPs** field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see [Configuring PAS](#).

6. In the **Certificates and Private Keys for HAProxy and Router** field, click the **Add** button to define one certificate keypair for HAProxy and Router. Since you have opted for unencrypted traffic behind the load balancer, then you can generate an RSA certificate in PAS.

7. In the **Minimum version of TLS supported by HAProxy and Router** field, select the minimum version of TLS to use in HAProxy communications. HAProxy uses TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the HAProxy, see [Cipher Suites](#).

8. Under **HAProxy forwards requests to Router over TLS**, select **Disable**.

9. If you want to use a specific set of TLS ciphers for HAProxy, configure **TLS Cipher Suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. Otherwise, leave the default values in this field.

10. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select **Disable SSL certificate verification for this environment**. Selecting this checkbox also disables SSL verification for route services.

    ![Use this checkbox only for development and testing environments. Do not select it for production environments.](image)

11. (Optional) If you do not want HAProxy or the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on HAProxy and Router** checkbox.

12. In the **Configure the CF Router support for the X-Forwarded-Client-Cert header** field, select **Always forward the XFCC header in the request**, regardless of the whether the client connection is mTLS.

13. Click **Save**.

14. After you complete the configuration in PCF, add your certificate or certificates to your load balancer, and configure its listening port. The procedures vary depending on your IaaS.

15. Configure your load balancer to append the **X-Forwarded-For** and **X-Forwarded-Proto** headers to client requests.

If the load balancer cannot be configured to provide the **X-Forwarded-Proto** header, the Gorouter will append it in requests forwarded to applications and system components, set to the IP address of the load balancer.

![Note: If the load balancer accepts unencrypted requests, it must provide the X-Forwarded-Proto header. Conversely, if the load balancer cannot be configured to send the X-Forwarded-Proto header, it should not accept unencrypted requests. Otherwise, applications and platform system components that require encrypted client requests will accept unencrypted requests when they should not accept them.](image)

### Terminating TLS at the Load Balancer and Gorouter

In this configuration two TLS connections are established: one from the client to the load balancer, and another from the load balancer to the Gorouter. This configuration secures all traffic between the load balancer and the Gorouter.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.

![TLS Termination at Load Balancer and Router](image)

This option is less performant, but allows for termination at a load balancer, as well as secure traffic between the load balancer and the Gorouter.

### Certificate Guidelines

In this deployment scenario, the following guidelines apply:
Certificates for the PAS domains must be stored on the load balancer, as well as on the Gorouter.

Generate certificates for your load balancer and the Gorouter with different keys. If the key for the certificate on the Gorouter is compromised, then the certificate on the load balancer is not at risk, and vice versa.

If you choose to host only one certificate on the Gorouter and many on your load balancer, configure your load balancer with the CA and hostname with which to validate the certificate hosted by the Gorouter.

About Hostname Verification

Hostname verification between the load balancer and the Gorouter is unnecessary when the load balancer is already configured with the Gorouter's IP address to correctly route the request.

If the load balancer uses DNS resolution to route requests to the Gorouters, then you should enable hostname verification.

About HTTP Header Forwarding

If you terminate TLS at your load balancer, then you must configure the load balancer to append the X-Forwarded-For and X-Forwarded-Proto HTTP headers to requests it sends to the Gorouter.

If you terminate TLS at your load balancer but it does not support HTTP, such that it cannot append HTTP headers, a workaround exists. We recommend you use this workaround only if your load balancer does not accept unencrypted requests. Configure your load balancer to send the client IP address using the PROXY protocol, and enable PROXY in the Gorouter. As the X-Forwarded-Proto header will not be present, configure the Gorouter to force-set this header to 'HTTPS'.

For more information about HTTP headers in CF, see HTTP Headers. If you are configuring the forwarding of client certificates, see Forward Client Certificate to Applications.

Procedure: Load Balancer and Gorouter

Perform the following steps to configure SSL termination on the Gorouter and load balancer in Pivotal Cloud Foundry (PCF):

1. Create an A record in your DNS that points to your load balancer IP address. The A record associates the System Domain and Apps Domain that you configure in the Domains section of the Pivotal Application Service (PAS) tile with the IP address of your load balancer.

   For example, with cf.example.com as the main subdomain for your Cloud Foundry (CF) deployment and a load balancer IP address 198.51.100.1, you must create an A record in your DNS that serves example.com and points *.cf to 198.51.100.1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.cf</td>
<td>A</td>
<td>198.51.100.1</td>
<td>example.com</td>
</tr>
</tbody>
</table>

2. Navigate to the Ops Manager Installation Dashboard.

3. Click the PAS tile in the Installation Dashboard.

4. Click Networking.

5. For PCF deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the Router IPs field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see Configuring PAS.

6. In the Certificates and Private Keys for HAProxy and Router field, click the Add button to define at least one certificate keypair for HAProxy and Router. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in PAS. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate for PCF Deployments.

7. In the Minimum version of TLS supported by HAProxy and Router, select the minimum version of TLS to use in HAProxy and Gorouter communications. The Gorouter use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites.

8. If you are using HAProxy, complete the following steps:
   a. Under HAProxy forwards requests to Router over TLS, select Enable.
   b. In the Certificate Authority for HAProxy Backend field, specify the Certificate Authority (CA) that signed the certificate you configured in the...
Certificate and Private Key for HAProxy and Router field.

If you used the Generate RSA Certificate link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate at the /api/v0/certificateAuthorities endpoint in the Ops Manager API.

c. If you want to use a specific set of TLS ciphers for HAProxy, configure TLS Cipher Suites for HAProxy. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. Otherwise, leave the default values in this field.
d. In the Configure the CF Router support for the X-Forwarded-Client-Cert header field, select Always forward the XFCC header in the request, regardless of whether the client connection is mTLS.
e. Proceed to step 11.

9. If you want to use a specific set of TLS ciphers for the Gorouter, configure TLS Cipher Suites for Router. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites. Otherwise, leave the default values in this field.

10. If you are not using HAProxy, complete the following steps:

  a. Under HAProxy forwards requests to Router over TLS, select Disable.
  b. In the Configure the CF Router support for the X-Forwarded-Client-Cert header field, select any of the available options depending on your client application needs. For more information about XFCC header forwarding, see Forward Client Certificate to Applications.
  c. In the PAS tile, click Resource Config.
  d. In the Instances drop down for the HAPProxy job, select 0 instances.
  e. Click Save.

11. (Optional) If you are not using SSL encryption or if you are using self-signed certificates, you can select Disable SSL certificate verification for this environment. Selecting this checkbox also disables SSL verification for route services.

   Use this checkbox only for development and testing environments. Do not select it for production environments.

12. (Optional) If you do not want HAProxy or the Gorouter to accept any non-encrypted HTTP traffic, select the Disable HTTP on HAProxy and Router checkbox.

13. Click Save.

14. After you complete the configuration in PCF, add your certificate or certificates to your load balancer, and configure its listening port. The procedures vary depending on your IaaS.

15. Configure your load balancer to append the X-Forwarded-For and X-Forwarded-Proto headers to client requests.

   If you cannot configure the load balancer to provide the X-Forwarded-For header, the Gorouter appends it in requests forwarded to applications and system components, set to the IP address of the load balancer.

   **Note:** If the load balancer accepts unencrypted requests, it must provide the X-Forwarded-Proto header. Conversely, if the load balancer cannot be configured to send the X-Forwarded-Proto header, it should not accept unencrypted requests. Otherwise, applications and platform system components that require encrypted client requests will accept unencrypted requests when they should not accept them.
**Enabling TCP Routing**

This topic describes enabling TCP Routing for your Cloud Foundry (CF) deployment. This feature enables developers to run applications that serve requests on non-HTTP TCP protocols. You can use TCP Routing to comply with regulatory requirements that require your organization to terminate the TLS as close to your apps as possible so that packets are not decrypted before reaching the application level.

**Route Ports**

The diagram below shows the layers of network address translation that occur in Cloud Foundry in support of TCP Routing. The descriptions step through an example work flow that covers route ports, backend ports, and app ports.

- A developer creates a TCP route for their application based on a TCP domain and a route port, and maps this route to one or more applications. See the Creating Routes topic for more information.
- Clients make requests to the route. DNS resolves the domain name to the load balancer.
- The load balancer listens on the port and forwards requests for the domain to the TCP routers. The load balancer must listen on a range of ports to support multiple TCP route creation. Additionally, Cloud Foundry must be configured with this range, so that the platform knows what ports can be reserved when developers create TCP routes.
- The TCP router can be dynamically configured to listen on the port when the route is mapped to an application. The domain the request was originally sent to is no longer relevant to the routing of the request to the application. The TCP router keeps a dynamically updated record of the backends for each route port. The backends represent instances of an application mapped to the route. The TCP Router chooses a backend using a round-robin load balancing algorithm for each new TCP connection from a client. As the TCP Router is protocol agnostic, it does not recognize individual requests, only TCP connections. All client requests transit the same connection to the selected backend until the client or backend closes the connection. Each subsequent connection triggers the selection of a backend.
- Because containers each have their own private network, the TCP router does not have direct access to application containers. When a container is created for an application instance, a port on the Cell VM is randomly chosen and iptables are configured to forward requests for this port to the internal interface on the container. The TCP router then receives a mapping of the route port to the Cell IP and port.
- By default, the Diego Cell only routes requests to port 8080, the App Port, on the container internal interface. The App Port is the port on which applications must listen. Developers can use the Cloud Controller API to update the ports an app can receive requests on. For more information, see Configuring Apps to Listen on Custom Ports (Beta).

**Pre-Deployment Steps**

**Note:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This disables TCP routing.

Before enabling TCP Routing, you must complete the following steps to set up networking requirements.

1. Choose a domain name from which your developers will create TCP routes for their applications. For example, create a domain which is similar to your app domain but prefixed by the TCP subdomain: tcp.APPS-DOMAIN.com

2. Configure DNS to resolve this domain name to the IP address of a highly-available load balancer that will forward traffic for the domain to the TCP routers. For more information, view the Domains topic. If you are operating an environment that does not require high-availability, configure DNS to
resolve the TCP domain name you have chosen directly to a single instance of the TCP Router.

3. (Optional) Choose IP addresses for the TCP routers and configure your load balancer to forward requests for the domain you chose in the step above to these addresses. Skip this step if you have configured DNS to resolve the TCP domain name to an instance of the TCP Router. Review the Enable TCP Routing steps in the Configuring PAS topic to configure your IP addresses for your PCF deployment.

4. (Optional) Decide on the number of TCP routes you want to support. For each TCP route, you must reserve a port. Configure your load balancer to forward the range of ports to the TCP routers. Skip this step if you have configured DNS to resolve the TCP domain name to an instance of the TCP Router.

5. Review the “Enable TCP Routing” steps in the Configuring PAS topic for your IaaS to configure your ports for your PCF deployment.

Post-Deployment Steps

In the following steps you use the Cloud Foundry Command Line Interface (cf CLI) to add the TCP shared domain and configure organization quotas to grant developers the ability to create TCP routes. This requires an admin user account.

Configure CF with Your TCP Domain

After deployment, you must configure Cloud Foundry with the domain that you configured in the Pre-Deployment step above so that developers can create TCP routes from it.

1. Run `cf router-groups`. You should see `default-tcp` as a response.

2. Run `cf create-shared-domain` to create a shared domain and associate it with the router group.

   ```
   $ cf create-shared-domain tcp.APPS-DOMAIN.com --router-group default-tcp
   ```

3. Run `cf domains`. Verify that next to your TCP domain, `TCP` appears under `type`.

Configure a Quota for TCP Routes

Since TCP route ports are a limited resource in some environments, quotas are configured to allow creation of zero TCP routes by default. After you deploy Cloud Foundry, you can increase the maximum number of TCP routes for all organizations or for particular organizations and spaces. Because you reserve a route port for each TCP route, the quota for this resource is managed with the cf CLI command option `--reserved-route-ports`. See the Creating and Modifying Quota Plans topic for more information.

If you have a default quota that applies to all organizations, you can update it to configure the number of route ports that can be reserved by each organization.

```
$ cf update-quota QUOTA --reserved-route-ports X
```

To create a new organization quota that can be allocated to particular organizations, provide the following required quota attributes in addition to the number of reserved route ports:

```
$ cf create-quota QUOTA --reserved-route-ports X
```

You can also create a quota that governs the number of route ports that can be created in a particular space.

```
$ cf create-space-quota QUOTA --reserved-route-ports X
```

Create a TCP Route

For instructions about creating a TCP Route, see the Create a TCP Route with a Port topic.
Router Groups

In Post-Deployment Steps, we describe that in order to create a domain from which to create TCP routes, it must be associated with the TCP Router Group. A router group is a cluster of identically configured routers. Router Groups were introduced as mechanism to support reservation of the same port for multiple TCP routes, thus increasing the capacity for TCP routes. However, only one router group is currently supported. In the Pre-Deployment Steps we describe how an admin user can configure the port range available for TCP routes in preparation for deployment.

Modify your TCP ports

After deployment, you can modify the range of ports available for TCP routes using `cf curl` commands, as demonstrated with the commands below. These commands require an admin user with the `routing.router_groups.read` and `routing.router_groups.write` scopes.

1. In a terminal window, run `cf curl /routing/v1/router_groups` to view the `reservable_ports`:

```
$ cf curl /routing/v1/router_groups
[
  {
    "guid": "9d1c01d9-9cb6-45f8-45c8-256b8579455c",
    "name": "default-tcp",
    "type": "tcp",
    "reservable_ports": "60000-60098"
  }
]
```

2. Modify the `reservable_ports`:

```
$ cf curl /routing/v1/router_groups/f7392031-a488-4890-8835-c4a038a3bded -X PUT -d '{"reservable_ports":"1024-1199"}'
```
Service Mesh (Beta)

This topic describes service mesh for Pivotal Application Service (PAS).

To deploy service mesh, see Deploying Service Mesh (Beta).

Overview

PAS includes an optional, beta routing plane that uses a service mesh. A service mesh provides traffic management, security, and observability for microservices. For more information, see What is a service mesh? in the Istio documentation.

Service mesh in PAS uses Istio Pilot and Envoy. The Cloud Foundry istio-release packages these components into a BOSH release. For more information, see the following:

- The Pilot section in Istio documentation.
- The What is Envoy topic in the Envoy documentation.
- The istio-release repository in GitHub.

Service mesh deploys an additional router and runs as a parallel routing plane as illustrated in the following diagram:

![Service Mesh Diagram]

Features

Service mesh currently supports configuring routing weights for app. For more information, see Using Weighted Routing (Beta).

Limitations

Consider the following when deploying service mesh:

- It does not have feature parity with the existing routing plane in PAS.
- It is for deployments with fewer than 20,000 routes. At greater scale, it may impact core platform functions.
- The control plane is not HA and registration of new routes may be delayed during upgrade.
- The domain for routes is *.mesh.YOUR-APPS-DOMAIN and is not configurable.

Component VMs

The following table describes each component VM deployed as part of service mesh in PAS, along with their function.

<table>
<thead>
<tr>
<th>VM</th>
<th>Processes</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>istio-router</td>
<td>envoy</td>
<td>A reverse proxy to forward HTTP/HTTPS requests external to the platform to applications on the platform.</td>
</tr>
<tr>
<td>istio-control</td>
<td>copilot, pilot-discovery</td>
<td>Propagates PAS external routes to all service mesh routers.</td>
</tr>
</tbody>
</table>
route-syncer  cc-route-syncer  Syncs routes created through the Cloud Controller API to the service mesh control plane.

For information about the communication ports and protocols used by these components, see Service Mesh in the Routing Network Communications topic.
Supporting WebSockets

This topic explains how Cloud Foundry (CF) uses WebSockets, why developers use WebSockets in their applications, and how operators can configure their load balancer to support WebSockets.

Operators who use a load balancer to distribute incoming traffic across CF router instances must configure their load balancer for WebSockets. Otherwise, the Loggregator system cannot stream application logs to developers, or application event data and component metrics to third-party aggregation services. Additionally, developers cannot use WebSockets in their applications.

Understand WebSockets

The WebSocket protocol provides full-duplex communication over a single TCP connection. Applications can use WebSockets to perform real-time data exchange between a client and a server more efficiently than HTTP.

CF uses WebSockets for the following metrics and logging purposes:

1. To stream all application event data and component metrics from the Doppler server instances to the Traffic Controller.
2. To stream application logs from the Traffic Controller to developers using the Cloud Foundry Command Line Interface (cf CLI) or Apps Manager.
3. To stream all application event data and component metrics from the Traffic Controller over the Firehose endpoint to external applications or services.

For more information about these Loggregator components, see the Overview of the Loggregator System topic.

Configure Your Load Balancer for WebSockets

To form a WebSocket connection, the client sends an HTTP request that contains an Upgrade header and other headers required to complete the WebSocket handshake. You must configure your load balancer to not upgrade the HTTP request, but rather to pass the Upgrade header through to the CF router. The procedures required to configure your load balancer depends on your IaaS and load balancer. The following list includes several possible approaches:

- Some load balancers can recognize the Upgrade header and pass these requests through to the CF router without returning the WebSocket handshake response. This may or may not be default behavior, and may require additional configuration.
- Some load balancers do not support passing WebSocket handshake requests containing the Upgrade header to the CF router. For instance, the Amazon Web Services (AWS) Elastic Load Balancer (ELB) does not support this behavior. In this scenario, you must configure your load balancer to forward TCP traffic to your CF router to support WebSockets. If your load balancer does not support TCP pass-through of WebSocket requests on the same port as other HTTP requests, you can do one of the following:
  - Configure your load balancer to listen on a non-standard port (the built-in CF load balancer listens on 8443 by default for this purpose), and forward requests for this port in TCP mode. Application clients must make WebSockets upgrade requests to this port. If you choose this strategy, you must follow the steps below in the Set Your Loggregator Port section of this topic.
  - If a non-standard port is not acceptable, add a load balancer that will handle WebSocket traffic (or another IP on an existing load balancer) and configure it to listen on standard ports 80 and 443 in TCP mode. Configure DNS with a new hostname, such as ws.cf.example.com, to be used for WebSockets. This hostname should resolve to the new load balancer interface.

Note: Regardless of your IaaS and configuration, you must configure your load balancer to send the X-Forwarded-For and X-Forwarded-Proto headers for non-WebSocket HTTP requests on ports 80 and 443. See the Securing Traffic into Cloud Foundry topic for more information.

Set Your Loggregator Port

By default, PCF assigns port 443 for TCP/WebSocket communications. If you have configured your load balancer to use a port other than 443 for TCP/WebSocket traffic, you must edit the Loggregator Port field in the Networking pane of the PAS tile.
<table>
<thead>
<tr>
<th>HTTP Headers to Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAPerxy Request Max Buffer Size</td>
</tr>
<tr>
<td>HAPerxy Protected Domains</td>
</tr>
<tr>
<td>HAPerxy Trusted CIDs</td>
</tr>
<tr>
<td>Loggregator Port</td>
</tr>
</tbody>
</table>
Configuring Load Balancer Healthchecks for Cloud Foundry Routers

This topic describes how to configure load balancer healthchecks for Cloud Foundry (CF) routers to ensure that the load balancer only forwards requests to healthy router instances. You can also configure a healthcheck for your HAProxy if your deployment uses the HAProxy component.

In environments that require high availability, operators must configure their own redundant load balancer to forward traffic directly to the CF routers. In environments that do not require high availability, operators can skip the load balancer and configure DNS to resolve the CF domains directly to a single instance of a router.

Add Healthcheck Endpoints for Routers

Configure your load balancer to use the following HTTP healthcheck endpoints. Add the IP addresses of all router instances along with their corresponding port and path.

- HTTP Router (Gorouter): `http://GOROUTER_IP:8080/health`
- TCP Router: `http://TCP_ROUTER_IP:80/health`

The configuration above assumes the default healthcheck ports for the CF routers. To modify these ports, see the sections below.

Add a Healthcheck Endpoint for HAProxy

If you have deployed one or more instances of HAProxy between your infrastructure load balancer and Gorouters, configure your infrastructure load balancer to use the following HTTP healthcheck endpoint: `http://HAPROXY_IP:8080/health`.

The HAProxy is an optional component that provides some features that Gorouter does not and can be helpful for demonstrating horizontal scalability of the CF routers in environments where an infrastructure load balancer is not available.

Set the Healthy and Unhealthy Threshold Properties for the Gorouter

To maintain high availability during upgrades to the HTTP router, each router is upgraded on a rolling basis. During upgrade of a highly available environment with multiple routers, each router is shutdown, upgraded, and restarted before the next router is upgraded. This ensures that any pending HTTP request passed to the HTTP router are handled correctly.

Pivotal Application Service (PAS) uses the following properties:

- **Unhealthy Threshold**: Specifies the amount of time, in seconds, that the Router continues to accept connections before shutting down. During this period, the healthcheck reports `unhealthy` to cause load balancers to fail over to other routers. You should set this value greater than or equal to the maximum amount of time it could take your load balancer to consider a router instance unhealthy, given contiguous failed healthchecks.

- **Healthy Threshold**: Specifies the amount of time, in seconds, to wait until declaring the router instance started. This allows an external load balancer time to register the instance as `healthy`.

You can configure these properties from the **Settings > Network** tab.

The image and table below describe the behavior of the load balancer health checks when a router shuts down and is restarted.
A shutdown request is sent to the router.

The router receives shutdown request, which causes the following:
- The router begins sending Service Unavailable responses to the load balancer health checks.
- The load balancer continues sending HTTP request to the router

The load balancer considers the router to be in an unhealthy state, which causes the load balancer to stop sending HTTP requests to the router.

The time between step 2 and 3 is defined by the values of the health check interval and threshold configured on the load balancer.

The router shuts down. The interval between step 2 and 4 is defined by the Unhealthy Threshold property of the Gorouter. In general, the value of this property should be longer than the value of the interval and threshold values (interval x threshold) of the load balancer. This additional interval ensures that any remaining HTTP requests are handled before the router shuts down.

If the router shutdown is initiated by an upgrade, the Gorouter software is upgraded.

The router restarts. The router will return Service Unavailable responses for load balancer health checks for 20 seconds; during this time the routing table is preloaded.

The routers begins returning Service Available responses to the load balancer health check.

The load balancer considers the router to be in a healthy state. The time between step 7 and 8 is specified by the health check interval and threshold configured for your load balancer (health check threshold x health check interval).

Shutdown and upgrade of the other router begins.
Monitoring PAS

This guide describes how Pivotal Cloud Foundry (PCF) operators can monitor their Pivotal Application Service (PAS) deployments. For information about monitoring Pivotal Container Service (PKS) deployments, see Logging and Monitoring PKS.

For more information about logging and metrics in PCF, see Overview of Logging and Metrics.

Overview

This guide includes the following topics:

- **Key Performance Indicators**: A list of Key Performance Indicators (KPIs) that operators may want to monitor with their PAS deployment to help ensure it is in a good operational state.
- **Key Capacity Scaling Indicators**: A list of capacity scaling indicators that operators may want to monitor to determine when they need to scale their PAS deployments.
- **Selecting and Configuring a Monitoring System**: Guidance for setting up PAS with monitoring platforms to continuously monitor component metrics and trigger health alerts.

KPI and Scaling Changes from PAS v2.4 to v2.5

This table highlights new and changed KPIs and scaling indicators in PAS v2.5.

<table>
<thead>
<tr>
<th>Status</th>
<th>Metric</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
<td>CF Syslog Drain Bindings Count</td>
<td>In PAS v2.5, the CF Syslog Drain Bindings Count scaling indicator uses the following derived metric:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cf-syslog-drain.drain_adapter.drain_bindings / number of Syslog Adapters</td>
<td>CF Syslog Drain Bindings Count</td>
</tr>
<tr>
<td>New</td>
<td>Log Cache Caching Duration</td>
<td>The Log Cache Caching Duration scaling indicator indicates the age in milliseconds of the oldest data point stored in Log Cache. This metric is useful for determining when to scale the VMs on which Log Cache runs.</td>
<td>Log Cache Caching Duration</td>
</tr>
</tbody>
</table>

Monitor PCF Services

For information about KPIs and metrics for PCF services, see the following topics:

- **MySQL for PCF**: Monitoring MySQL for PCF.
- **Pivotal Cloud Cache**: See Monitoring Pivotal Cloud Cache.
- **Redis for PCF**: Monitoring Redis for PCF.
- **RabbitMQ for PCF (pre-provisioned)**: See Monitoring and KPIs for Pre-Provisioned RabbitMQ for PCF.
- **RabbitMQ for PCF (on-demand)**: See Monitoring and KPIs for On-Demand RabbitMQ for PCF.
Key Performance Indicators

This topic describes Key Performance Indicators (KPIs) that operators may want to monitor with their Pivotal Application Service (PAS) deployment to help ensure it is in a good operational state.

The following PAS v2.4 KPIs are provided for operators to give general guidance on monitoring a PAS deployment using platform component and system (BOSH) metrics. Although many metrics are emitted from the platform, the following PAS v2.4 KPIs are high-signal-value metrics that can indicate emerging platform issues.

This alerting and response guidance has been shown to apply to most deployments. Pivotal recommends that operators continue to fine-tune the alert measures to their deployment by observing historical trends. Pivotal also recommends that operators expand beyond this guidance and create new, deployment-specific monitoring metrics, thresholds, and alerts based on learning from their deployments.

Note: Thresholds noted as “dynamic” in the tables below indicate that while a metric is highly important to watch, the relative numbers to set threshold warnings at are specific to a given PAS deployment and its use cases. These dynamic thresholds should be occasionally revisited because the PCF foundation and its usage continue to evolve. See Determine Warning and Critical Thresholds for more information.

Diego Auctioneer Metrics

Auctioneer App Instance (AI) Placement Failures

<table>
<thead>
<tr>
<th>Description</th>
<th>auctioneer.AuctioneerLRPAuctionsFailed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The number of Long Running Process (LRP) instances that the auctioneer failed to place on Diego cells. This metric is cumulative over the lifetime of the auctioneer job.</td>
</tr>
<tr>
<td>Use</td>
<td>This metric can indicate that PAS is out of container space or that there is a lack of resources within your environment. This indicator also increases when the LRP is requesting an isolation segment, volume drivers, or a stack that is unavailable, either not deployed or lacking sufficient resources to accept the work.</td>
</tr>
<tr>
<td>This metric is emitted on event, and therefore gaps in receipt of this metric can be normal during periods of no app instances being scheduled.</td>
<td></td>
</tr>
<tr>
<td>This error is most common due to capacity issues, for example, if cells do not have enough resources, or if cells are going back and forth between a healthy and unhealthy state.</td>
<td></td>
</tr>
<tr>
<td>Origin</td>
<td>Firehose</td>
</tr>
<tr>
<td>Type</td>
<td>Counter (Integer)</td>
</tr>
<tr>
<td>Frequency</td>
<td>During each auction</td>
</tr>
<tr>
<td>Recommended measurement</td>
<td>Per minute delta averaged over a 5-minute window</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
<td>Yellow warning: ≥ 0.5</td>
</tr>
<tr>
<td></td>
<td>Red critical: ≥ 1</td>
</tr>
<tr>
<td>Recommended response</td>
<td>1. To best determine the root cause, examine the Auctioneer logs. Depending on the specific error and resource constraint, you may also find a failure reason in the Cloud Controller (CC) API.</td>
</tr>
<tr>
<td></td>
<td>2. Investigate the health of your Diego cells to determine if they are the resource type causing the problem.</td>
</tr>
<tr>
<td></td>
<td>3. Consider scaling additional cells using Ops Manager.</td>
</tr>
<tr>
<td></td>
<td>4. If scaling cells does not solve the problem, pull Diego brain logs and BBS node logs and contact Pivotal Support telling them that LRP auctions are failing.</td>
</tr>
</tbody>
</table>

Auctioneer Time to Fetch Cell State

<table>
<thead>
<tr>
<th>Description</th>
<th>auctioneer.AuctioneerFetchStatesDuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: Thresholds noted as “dynamic” in the tables below indicate that while a metric is highly important to watch, the relative numbers to set threshold warnings at are specific to a given PAS deployment and its use cases. These dynamic thresholds should be occasionally revisited because the PCF foundation and its usage continue to evolve. See Determine Warning and Critical Thresholds for more information.</td>
<td></td>
</tr>
</tbody>
</table>
### auctioneer.AuctioneerFetchStatesDuration

<table>
<thead>
<tr>
<th>Description</th>
<th>Time in ns that the auctioneer took to fetch state from all the Diego cells when running its auction.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use</strong></td>
<td>Indicates how the cells themselves are performing. Alerting on this metric helps alert that app staging requests to Diego may be failing.</td>
</tr>
<tr>
<td><strong>Origin</strong></td>
<td>Firehose</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Gauge, integer in ns</td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
<td>During event, during each auction</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended measurement</th>
<th>Maximum over the last 5 minutes divided by 1,000,000,000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Recommended alert thresholds</th>
<th>Yellow warning: ≥ 2 s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red critical: ≥ 5 s</td>
</tr>
</tbody>
</table>

#### Recommended response

1. Check the health of the cells by reviewing the logs and looking for errors.
2. Review IaaS console metrics.
3. Inspect the Auctioneer logs to determine if one or more cells is taking significantly longer to fetch state than other cells. Relevant log lines will have wording like ‘fetched cell state’.
4. Pull Diego brain logs, cell logs, and auctioneer logs and contact Pivotal Support telling them that fetching cell states is taking too long.

### Auctioneer App Instance Starts

<table>
<thead>
<tr>
<th>auctioneer.AuctioneerLRPAuctionsStarted</th>
</tr>
</thead>
</table>

#### Description

The number of LRP instances that the auctioneer successfully placed on Diego cells. This metric is cumulative over the lifetime of the auctioneer job.

**Use**: Provides a sense of running system activity levels in your environment. Can also give you a sense of how many app instances have been started over time. The recommended measurement, below, can help indicate a significant amount of container churn. However, for capacity planning purposes, it is more helpful to observe deltas over a long time window.

This metric is emitted on event, and therefore gaps in receipt of this metric can be normal during periods of no app instances being scheduled.

**Origin**: Firehose

**Type**: Counter (Integer)

**Frequency**: During event, during each auction

<table>
<thead>
<tr>
<th>Recommended measurement</th>
<th>Per minute delta averaged over a 5-minute window</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Recommended alert thresholds</th>
<th>Yellow warning: Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Red critical: Dynamic</td>
</tr>
</tbody>
</table>

#### Recommended response

When observing a significant amount of container churn, do the following:

1. Look to eliminate explainable causes of temporary churn, such as a deployment or increased developer activity.
2. If container churn appears to continue over an extended period, pull logs from the Diego Brain and BBS node before contacting Pivotal support.

When observing extended periods of high or low activity trends, scale up or down CF components as needed.

### Auctioneer Task Placement Failures

<table>
<thead>
<tr>
<th>auctioneer.AuctioneerTaskAuctionsFailed</th>
</tr>
</thead>
</table>

The number of Tasks that the auctioneer failed to place on Diego cells. This metric is cumulative over the lifetime
**Description**

of the auctioneer job.

Use: Failing Task auctions indicate a lack of resources within your environment and that you likely need to scale. This indicator also increases when the Task is requesting an isolation segment, volume drivers, or a stack that is unavailable, either not deployed or lacking sufficient resources to accept the work.

This metric is emitted on event, and therefore gaps in receipt of this metric can be normal during periods of no tasks being scheduled.

This error is most common due to capacity issues, for example, if cells do not have enough resources, or if cells are going back and forth between a healthy and unhealthy state.

Origin: Firehose

Type: Counter (Float)

Frequency: During event, each auction

---

**Recommended measurement**

Per minute delta averaged over a 5-minute window

---

**Recommended alert thresholds**

Yellow warning: ≥ 0.5

Red critical: ≥ 1

---

**Recommended response**

1. In order to best determine the root cause, examine the Auctioneer logs. Depending on the specific error or resource constraint, you may also find a failure reason in the CC API.

2. Investigate the health of Diego cells.

3. Consider scaling additional cells using Ops Manager.

4. If scaling cells does not solve the problem, pull Diego brain logs and BBS logs for troubleshooting and contact Pivotal Support for additional troubleshooting. Inform Pivotal Support that Task auctions are failing.

---

**Diego BBS Metrics**

**BBS Time to Run LRP Convergence**

<table>
<thead>
<tr>
<th>bbs.ConvergenceLRPDuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Time in ns that the BBS took to run its LRP convergence pass.</td>
</tr>
</tbody>
</table>

Use: If the convergence run begins taking too long, apps or Tasks may be crashing without restarting. This symptom can also indicate loss of connectivity to the BBS database.

Origin: Firehose

Type: Gauge (Integer in ns)

Frequency: During event, every 30 seconds when LRP convergence runs, emission should be near-constant on a running deployment

---

**Recommended measurement**

Maximum over the last 15 minutes divided by 1,000,000,000

---

**Recommended alert thresholds**

Yellow warning: ≥ 10 s

Red critical: ≥ 20 s

---

**Recommended response**

1. Check BBS logs for errors.

2. Try vertically scaling the BBS VM resources up. For example, add more CPUs or memory depending on its `system.cpu` and `system.memory` metrics.

3. Consider vertically scaling the PAS backing database, if `system.cpu` and `system.memory` metrics for the database instances are high.

4. If that does not solve the issue, pull the BBS logs and contact Pivotal Support for additional troubleshooting.
### BBS Time to Handle Requests

<table>
<thead>
<tr>
<th>Description</th>
<th>The maximum observed latency time over the past 60 seconds that the BBS took to handle requests across all its API endpoints. Diego is now aggregating this metric to emit the max value observed over 60 seconds. Use: If this metric rises, the PAS API is slowing. Response to certain cf CLI commands is slow if request latency is high. Origin: Firehose Type: Gauge (Integer in ns) Frequency: 60 s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recommended measurement</strong></td>
<td>Average over the last 15 minutes divided by 1,000,000,000</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
<td>Yellow warning: $\geq 5$ s Red critical: $\geq 10$ s</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
<td>1. Check CPU and memory statistics in Ops Manager. 2. Check BBS logs for faults and errors that can indicate issues with BBS. 3. Try scaling the BBS VM resources up. For example, add more CPUs/memory depending on its <code>system.cpu</code> and <code>system.memory</code> metrics. 4. Consider vertically scaling the PAS backing database, if <code>system.cpu</code> and <code>system.memory</code> metrics for the database instances are high. 5. If the above steps do not solve the issue, collect a sample of the cell logs from the BBS VMs and contact Pivotal Support to troubleshoot further.</td>
</tr>
</tbody>
</table>

### Cloud Controller and Diego in Sync

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates if the <code>cf-apps</code> Domain is up-to-date, meaning that CF App requests from Cloud Controller are synchronized to <code>bbs.LRPsDesired</code> (Diego-desired AIs) for execution. - 1 means <code>cf-apps</code> Domain is up-to-date - No data received means <code>cf-apps</code> Domain is not up-to-date Use: If the <code>cf-apps</code> Domain does not stay up-to-date, changes requested in the Cloud Controller are not guaranteed to propagate throughout the system. If the Cloud Controller and Diego are out of sync, then apps running could vary from those desired. Origin: Firehose Type: Gauge (Float) Frequency: 30 s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recommended measurement</strong></td>
<td>Value over the last 5 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
<td>Yellow warning: N/A Red critical: $\neq 1$ The recommended threshold value represents a state where an up-to-date metric <code>1</code> has not been received for the entire 5-minute window.</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
<td>1. Check the BBS and Clock Global (Cloud Controller clock) logs. 2. If the problem continues, pull the BBS logs and Clock Global (Cloud Controller clock) logs and contact Pivotal Support to say that the <code>cf-apps</code> domain is not being kept fresh.</td>
</tr>
</tbody>
</table>
## More App Instances Than Expected

<table>
<thead>
<tr>
<th>bbs.LRPExtra</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Total number of LRP instances that are no longer desired but still have a BBS record. When Diego wants to add more apps, the BBS sends a request to the auctioneer to spin up additional LRPs. LRPExtra is the total number of LRP instances that are no longer desired but still have a BBS record. Use: If Diego has more LRPs running than expected, there may be problems with the BBS. Deleting an app with many instances can temporarily spike this metric. However, a sustained spike in <strong>bbs.LRPExtra</strong> is unusual and should be investigated.</td>
</tr>
<tr>
<td>Origin: Firehose Type: Gauge (Float) Frequency: 30 s</td>
</tr>
<tr>
<td><strong>Recommended measurement</strong></td>
</tr>
<tr>
<td>Average over the last 5 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
</tr>
<tr>
<td>Yellow warning: ≥ 5 Red critical: ≥ 10</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
</tr>
<tr>
<td>1. Review the BBS logs for proper operation or errors, looking for detailed error messages. 2. If the condition persists, pull the BBS logs and contact Pivotal Support.</td>
</tr>
</tbody>
</table>

## Fewer App Instances Than Expected

<table>
<thead>
<tr>
<th>bbs.LRPsMissing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Total number of LRP instances that are desired but have no record in the BBS. When Diego wants to add more apps, the BBS sends a request to the auctioneer to spin up additional LRPs. LRPsMissing is the total number of LRP instances that are desired but have no BBS record. Use: If Diego has less LRP running than expected, there may be problems with the BBS. An app push with many instances can temporarily spike this metric. However, a sustained spike in <strong>bbs.LRPsMissing</strong> is unusual and should be investigated.</td>
</tr>
<tr>
<td>Origin: Firehose Type: Gauge (Float) Frequency: 30 s</td>
</tr>
<tr>
<td><strong>Recommended measurement</strong></td>
</tr>
<tr>
<td>Average over the last 5 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
</tr>
<tr>
<td>Yellow warning: ≥ 5 Red critical: ≥ 10</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
</tr>
<tr>
<td>1. Review the BBS logs for proper operation or errors, looking for detailed error messages. 2. If the condition persists, pull the BBS logs and contact Pivotal Support.</td>
</tr>
</tbody>
</table>

## Crashed App Instances

<table>
<thead>
<tr>
<th>bbs.CrashedActualLRPs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Total number of LRP instances that have crashed. Use: Indicates how many instances in the deployment are in a crashed state. An increase in <strong>bbs.CrashedActualLRPs</strong> can indicate several problems, from a bad app with many instances associated, to a platform issue that is resulting in app crashes. Use this metric to help create a baseline for your deployment. After you have a baseline, you can create a deployment-specific alert to notify of a spike in crashes above the...</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019

959 2.5
Running App Instances, Rate of Change

**1hr average of bbs.LRPsRunning – prior 1hr average of bbs.LRPsRunning**

<table>
<thead>
<tr>
<th>Description</th>
<th>Rate of change in app instances being started or stopped on the platform. It is derived from and represents the total number of LRP instances that are running on Diego cells. Use: Delta reflects upward or downward trend for app instances started or stopped. Helps to provide a picture of the overall growth trend of the environment for capacity planning. You may want to alert on delta values outside of the expected range.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Firehose</td>
</tr>
<tr>
<td>Type</td>
<td>Gauge (Float)</td>
</tr>
<tr>
<td>Frequency</td>
<td>During event, emission should be constant on a running deployment</td>
</tr>
</tbody>
</table>

**Recommended measurement**

- **derived=(1-hour average of bbs.LRPsRunning - prior 1-hour average of bbs.LRPsRunning)**

**Recommended alert thresholds**

- **Yellow warning:** Dynamic
- **Red critical:** Dynamic

**Recommended response**

1. Look at the BBS logs for apps that are crashing and at the cell logs to see if the problem is with the apps themselves, rather than a platform issue.
2. Before contacting Pivotal Support, pull the BBS logs and, if particular apps are the problem, pull the logs from their Diego cells too.

Diego Cell Metrics

Remaining Memory Available — Cell Memory Chunks Available

<table>
<thead>
<tr>
<th>rep.CapacityRemainingMemory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining amount of memory in MiB available for this Diego cell to allocate to containers. Use: Indicates the available cell memory. Insufficient cell memory can prevent pushing and scaling apps.</td>
</tr>
<tr>
<td>The strongest operational value of this metric is to understand a deployment's average app size and monitor/alert on ensuring that at least some cells have large enough capacity to accept standard app size pushes. For example, if pushing a 4 GB app, Diego would have trouble placing that app if there is no one cell with sufficient capacity of 4 GB or greater.</td>
</tr>
<tr>
<td>As an example, Pivotal Cloud Ops uses a standard of 4 GB, and computes and monitors for the number of cells with at least 4 GB free. When the number of cells with at least 4 GB falls below a defined threshold, this is a scaling indicator alert to increase capacity. This free chunk count threshold should be tuned to the deployment size and the standard size of apps being pushed to the deployment.</td>
</tr>
<tr>
<td>Origin: Firehose</td>
</tr>
</tbody>
</table>
### Recommended measurement

- **Type:** Gauge (Integer in MiB)
- **Frequency:** 60 s

For alerting:

1. Determine the size of a standard app in your deployment. This is the suggested value to calculate free chunks of Remaining Memory by.

2. Create a script/tool that can iterate through each Diego Cell and do the following:
   
   a. Pull the `rep.CapacityRemainingMemory` metric for each cell.
   b. Divide the values received by 1000 to get the value in Gigabytes (if desired threshold is GB-based).
   c. Compare recorded values to your minimum capacity threshold, and count the number of cells that have equal or greater than the desired amount of free chunk space.

3. Determine a desired scaling threshold based on the minimum amount of free chunks that are acceptable in this deployment given historical trends.

4. Set an alert to indicate the need to scale cell memory capacity when the value falls below the desired threshold number.

For visualization purposes:

Looking at this metric (`rep.CapacityRemainingMemory`) as a minimum value per cell has more informational value than alerting value. It can be an interesting heatmap visualization, showing average variance and density over time.

### Recommended alert thresholds

- **Yellow warning:** Dynamic
- **Red critical:** Dynamic

### Recommended response

1. Assign more resources to the cells or assign more cells.
2. Scale additional Diego cells using Ops Manager.

### Alternative Metric

If you are using PCF Healthwatch, Pivotal recommends the metric `healthwatch.Diego.AvailableFreeChunks` for this purpose.

---

### Remaining Memory Available — Overall Remaining Memory Available

<table>
<thead>
<tr>
<th>Description</th>
<th>Remaining amount of memory in MiB available for this Diego cell to allocate to containers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use:</td>
<td>Can indicate low memory capacity overall in the platform. Low memory can prevent app scaling and new deployments. The overall sum of capacity can indicate that you need to scale the platform. Observing capacity consumption trends over time helps with capacity planning.</td>
</tr>
<tr>
<td>Origin:</td>
<td>Firehose</td>
</tr>
<tr>
<td>Type:</td>
<td>Gauge (Integer in MiB)</td>
</tr>
<tr>
<td>Frequency:</td>
<td>60 s</td>
</tr>
</tbody>
</table>

#### Recommended measurement

- Minimum over the last 5 minutes divided by 1024 (across all instances)

#### Recommended alert thresholds

- **Yellow warning:** \( \leq 64 \text{ GB} \)
- **Red critical:** \( \leq 32 \text{ GB} \)

#### Recommended response

1. Assign more resources to the cells or assign more cells.
2. Scale additional Diego cells via Ops Manager.

#### Alternative Metric

If you are using PCF Healthwatch, Pivotal recommends the metric `healthwatch.Diego.AvailableFreeChunks` for this purpose.

---

### Remaining Disk Available — Cell Disk Chunks Available
### rep.CapacityRemainingDisk

**Description**

Remaining amount of disk in MiB available for this Diego cell to allocate to containers.

*Use:* Indicates the available cell disk. Insufficient free disk on cell(s) will prevent the staging or starting of apps or tasks, resulting in error messages like `ERR Failed to stage application: insufficient resources`.

Because Diego will fail to stage without at least 6 GB free, unreserved disk space on a given cell, the strongest operational value of this metric is to ensure that at least some cells have a large enough disk capacity to support the staging of apps and tasks.

Pivotal recommends computing and monitoring for the number of cells with at least 6 GB Disk free. When the number of cells with at least 6 GB falls below a defined threshold, this is a scaling indicator alert to increase capacity. The alerting threshold value for the amount of free chunks of Disk should be tuned to the deployment size and the standard size of apps being pushed to the deployment. *Origin: Firehose*

*Type:* Gauge (Integer in MiB)

*Frequency:* 60 s

**Recommended measurement**

For alerting:

1. Because Diego will fail to stage without at least 6 GB free, this is the suggested minimum value to calculate free chunks of Remaining Disk by.

2. Create a script/tool that can iterate through each Diego Cell and do the following:
   a. Pull the `rep.CapacityRemainingDisk` metric for each cell.
   b. Divide the values received by 1000 to get the value in Gigabytes (if desired threshold is GB-based).
   c. Compare recorded values to your minimum capacity threshold, and count the number of cells that have equal or greater than the desired amount of free chunk space.

3. Determine a desired scaling threshold based on the minimum amount of free chunks that are acceptable in this deployment given historical trends.

4. Set an alert to indicate the need to scale cell disk capacity when the value falls below the desired threshold number.

*For visualization purposes:* Looking at this metric (`rep.CapacityRemainingDisk`) as a minimum value per cell has more informational value than alerting value. It can be an interesting heatmap visualization, showing average variance and density over time.

**Recommended alert thresholds**

- **Yellow warning:** Dynamic
- **Red critical:** Dynamic

**Recommended response**

1. Assign more resources to the cells or assign more cells.

2. Scale additional Diego cells using Ops Manager.

**Alternative Metric**

If you are using PCF Healthwatch, Pivotal recommends the metric `healthwatch.Diego.AvailableFreeChunksDisk` for this purpose.

### Remaining Disk Available - Overall Remaining Disk Available

**Description**

Remaining amount of disk in MiB available for this Diego cell to allocate to containers.

*Use:* Low disk capacity can prevent app scaling and new deployments. Because Diego staging Tasks can fail without at least 6 GB free, the recommended red threshold is based on the minimum disk capacity across the deployment falling below 6 GB in the previous 5 minutes.

It can also be meaningful to assess how many chunks of free disk space are above a given threshold, similar to `rep.CapacityRemainingMemory`.

*Origin: Firehose*

*Type:* Gauge (Integer in MiB)
**Cell Rep Time to Sync**

<table>
<thead>
<tr>
<th>Description</th>
<th>rep.RepBulkSyncDuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in ns that the Diego Cell Rep took to sync the ActualLRPs that it claimed with its actual garden containers.</td>
<td></td>
</tr>
<tr>
<td>Use: Sync times that are too high can indicate issues with the BBS.</td>
<td></td>
</tr>
<tr>
<td>Origin: Firehose</td>
<td></td>
</tr>
<tr>
<td>Type: Gauge (Float in ns)</td>
<td></td>
</tr>
<tr>
<td>Frequency: 30 s</td>
<td></td>
</tr>
</tbody>
</table>

**Recommended measurement**

- Maximum over the last 15 minutes divided by 1,000,000,000

**Recommended alert thresholds**

- Yellow warning: ≥ 5 s
- Red critical: ≥ 10 s

**Recommended response**

1. Investigate BBS logs for faults and errors.
2. If a particular cell or cells appear problematic, pull logs for the cells and the BBS logs before contacting Pivotal Support.

**Unhealthy Cells**

<table>
<thead>
<tr>
<th>Description</th>
<th>rep.UnhealthyCell</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Diego cell periodically checks its health against the garden backend. For Diego cells, 0 means healthy, and 1 means unhealthy.</td>
<td></td>
</tr>
<tr>
<td>Use: Set an alert for further investigation if multiple unhealthy Diego cells are detected in the given time window. If one cell is impacted, it does not participate in auctions, but end-user impact is usually low. If multiple cells are impacted, this can indicate a larger problem with Diego, and should be considered a more critical investigation need.</td>
<td></td>
</tr>
<tr>
<td>Suggested alert threshold based on multiple unhealthy cells in the given time window.</td>
<td></td>
</tr>
<tr>
<td>Although end-user impact is usually low if only one cell is impacted, this should still be investigated. Particularly in a lower capacity environment, this situation could result in negative end-user impact if left unresolved.</td>
<td></td>
</tr>
<tr>
<td>Origin: Firehose</td>
<td></td>
</tr>
<tr>
<td>Type: Gauge (Float, 0-1)</td>
<td></td>
</tr>
<tr>
<td>Frequency: 30 s</td>
<td></td>
</tr>
</tbody>
</table>

**Recommended measurement**

- Maximum over the last 5 minutes

**Recommended alert thresholds**

- Yellow warning: ≥ 1
- Red critical: > 1

1. Investigate Diego cell servers for faults and errors.
2. If a particular cell or cells appear problematic:
Diego Locket Metrics

Active Locks

<table>
<thead>
<tr>
<th>Description</th>
<th>Total count of how many locks the system components are holding.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use:</td>
<td>If the ActiveLocks count is not equal to the expected value, there is likely a problem with Diego.</td>
</tr>
<tr>
<td>Origin:</td>
<td>Firehose</td>
</tr>
<tr>
<td>Type:</td>
<td>Gauge</td>
</tr>
<tr>
<td>Frequency:</td>
<td>60 s</td>
</tr>
</tbody>
</table>

Recommended measurement: Maximum over the last 5 minutes

Recommended alert thresholds:
- Yellow warning: N/A
- Red critical: ≠ 5

Recommended response:
1. Run `monit status` to inspect for failing processes.

2. If there are no failing processes, then review the logs for the components using the Locket service: BBS, Auctioneer, TPS Watcher, Routing API, and Clock Global (Cloud Controller clock). Look for indications that only one of each component is active at a time.

3. Focus triage on the BBS first:
   - A healthy BBS shows obvious activity around starting or claiming LRPs.
   - An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer.
   - Reference the BBS-level Locket metric `Locks Held by BBS`. A value of 0 indicates Locket issues at the BBS level.

4. If the BBS appears healthy, then check the Auctioneer to ensure it is processing auction payloads.
   - Recent logs for Auctioneer should show all but one of its instances are currently waiting on locks, and the active Auctioneer should show a record of when it last attempted to execute work. This attempt should correspond to app development activity, such as `cf push`.
   - Reference the Auctioneer-level Locket metric `Locks Held by Auctioneer`. A value of 0 indicates Locket issues at the Auctioneer level.

5. The TPS Watcher is primarily active when app instances crash. Therefore, if the TPS Watcher is suspected, review the most recent logs.

6. If you are unable to resolve on-going excessive active locks, pull logs from the Diego BBS and Auctioneer VMs, which includes the Locket service component logs, and contact Pivotal Support.

Locks Held by BBS

© Copyright Pivotal Software Inc, 2013-2019
### bbs.LockHeld

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whether a BBS instance holds the expected BBS lock (in Locket). 1 means the active BBS server holds the lock, and 0 means the lock was lost.</td>
</tr>
<tr>
<td>Use: This metric is complimentary to Active Locks, and it offers a BBS-level version of the Locket metrics. Although it is emitted per BBS instance, only 1 active lock is held by BBS. Therefore, the expected value is 1. The metric may occasionally be 0 when the BBS instances are performing a leader transition, but a prolonged value of 0 indicates an issue with BBS.</td>
</tr>
<tr>
<td>Origin: Firehose</td>
</tr>
<tr>
<td>Type: Gauge</td>
</tr>
<tr>
<td>Frequency: Periodically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum over the last 5 minutes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended alert thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow warning: N/A</td>
</tr>
<tr>
<td>Red critical: ≠ 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Run <code>monit status</code> on the Diego database VM to check for failing processes.</td>
</tr>
<tr>
<td>2. If there are no failing processes, then review the logs for BBS.</td>
</tr>
<tr>
<td>- A healthy BBS shows obvious activity around starting or claiming LRPs.</td>
</tr>
<tr>
<td>- An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer.</td>
</tr>
<tr>
<td>3. If you are unable to resolve the issue, pull logs from the Diego BBS, which include the Locket service component logs, and contact Pivotal Support.</td>
</tr>
</tbody>
</table>

### Locks Held by Auctioneer

<table>
<thead>
<tr>
<th>auctioneer.LockHeld</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whether an Auctioneer instance holds the expected Auctioneer lock (in Locket). 1 means the active Auctioneer holds the lock, and 0 means the lock was lost.</td>
</tr>
<tr>
<td>Use: This metric is complimentary to Active Locks, and it offers an Auctioneer-level version of the Locket metrics. Although it is emitted per Auctioneer instance, only 1 active lock is held by Auctioneer. Therefore, the expected value is 1. The metric may occasionally be 0 when the Auctioneer instances are performing a leader transition, but a prolonged value of 0 indicates an issue with Auctioneer.</td>
</tr>
<tr>
<td>Origin: Firehose</td>
</tr>
<tr>
<td>Type: Gauge</td>
</tr>
<tr>
<td>Frequency: Periodically</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum over the last 5 minutes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended alert thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow warning: N/A</td>
</tr>
<tr>
<td>Red critical: ≠ 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommended response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Run <code>monit status</code> on the Diego Database VM to check for failing processes.</td>
</tr>
<tr>
<td>2. If there are no failing processes, then review the logs for Auctioneer.</td>
</tr>
<tr>
<td>- Recent logs for Auctioneer should show all but one of its instances are currently waiting on locks, and the active Auctioneer should show a record of when it last attempted to execute work. This attempt should correspond to app development activity, such as <code>cf push</code>.</td>
</tr>
<tr>
<td>3. If you are unable to resolve the issue, pull logs from the Diego BBS and Auctioneer VMs, which includes the Locket service component logs, and contact Pivotal Support.</td>
</tr>
</tbody>
</table>

### Active Presences
**locket.ActivePresences**

| Description | Total count of active presences. Presences are defined as the registration records that the cells maintain to advertise themselves to the platform. Use: If the Active Presences count is far from the expected, there might be a problem with Diego. The number of active presences varies according to the number of cells deployed. Therefore, during purposeful scale adjustments to PAS, this alerting threshold should be adjusted. Establish an initial threshold by observing the historical trends for the deployment over a brief period of time, increase the threshold as more cells are deployed. During a rolling deploy, this metric shows variance during the BOSH lifecycle when cells are evacuated and restarted. Tolerable variance is within the bounds of the BOSH max inflight range for the instance group. Origin: Firehose Type: Gauge Frequency: 60 s |

**Recommended measurement** | Maximum over the last 15 minutes |

**Recommended alert thresholds** | Yellow warning: Dynamic Red critical: Dynamic |

**Recommended response**
1. Ensure that the variance is not the result of an active rolling deploy. Also ensure that the alert threshold is appropriate to the number of cells in the current deployment.
2. Run `monit status` to inspect for failing processes.
3. If there are no failing processes, then review the logs for the components using the Locket service itself on Diego BBS instances.
4. If you are unable to resolve the problem, pull the logs from the Diego BBS, which include the Locket service component logs, and contact Pivotal Support.

---

**Diego Route Emitter Metrics**

**Route Emitter Time to Sync**

| route_emitter.RouteEmitterSyncDuration | Time in ns that the active Route Emitter took to perform its synchronization pass. Use: Increases in this metric indicate that the Route Emitter may have trouble maintaining an accurate routing table to broadcast to the Gorouters. Tune alerting values to your deployment based on historical data and adjust based on observations over time. The suggested starting point is ≥ 5 for the yellow threshold and ≥ 10 for the critical threshold. Pivotal has observed on its Pivotal Web Services deployment that above 10 s, the BBS may be failing. Origin: Firehose Type: Gauge (Float in ns) Frequency: 60 s |

**Recommended measurement** | Maximum, per job, over the last 15 minutes divided by 1,000,000,000 |

**Recommended alert thresholds** | Yellow warning: Dynamic Red critical: Dynamic |

**Recommended response**
If all or many jobs showing as impacted, there is likely an issue with Diego.
1. Investigate the Route Emitter and Diego BBS logs for errors.
2. Verify that app routes are functional by making a request to an app, pushing an app and pinging it, or if applicable, checking that your smoke tests have passed.
If one or a few jobs showing as impacted, there is likely a connectivity issue and the impacted job should be investigated further.

PAS MySQL KPIs

When PAS uses an internal MySQL database, as configured in the PAS tile Settings tab > Databases pane, the database cluster generates KPIs as described below.

Note: This section assumes you are using the Internal Databases - MySQL - MariaDB Galera Cluster option as your system database.

MySQL Server Availability

<table>
<thead>
<tr>
<th>/mysql/available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Recommended measurement</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Recommended response</td>
</tr>
</tbody>
</table>

Galera Cluster Node Readiness

<table>
<thead>
<tr>
<th>/mysql/galera/wsrep_ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Recommended measurement</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Recommended response</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Galera Cluster Size

| /mysql/galera/wsrep_cluster_size |
The number of cluster nodes with which each node is communicating normally.

**Description**
Use: When running in a multi-node configuration, this metric indicates if each member of the cluster is communicating normally with all other nodes.

Origin: Firehose
Envelope Type: Gauge
Unit: count
Frequency: 30 s (default)

**Recommended measurement**
(Average of the values of each node / cluster size), over the last 5 minutes

**Recommended alert thresholds**
Yellow warning: < 3.0 (availability compromised)
Red critical: < 1.0 (cluster unavailable)

**Recommended response**
- Run `mysql-diag` and check the MySQL Server logs for errors.

---

**Galera Cluster Status**

**/mysql/galera/wsrep_cluster_status**

Shows the primary status of the cluster component that the node is in. Values are:
- Primary = 1
- Non-primary = 0
- Disconnected = -1


Origin: Firehose
Envelope Type: Gauge
Unit: integer (see above)
Frequency: 30 s (default)

**Recommended measurement**
Sum of each of the nodes, over the last 5 minutes

**Recommended alert thresholds**
Yellow warning: < 3
Red critical: < 1

**Recommended response**
- Check node status to ensure that they are all in working order and able to receive write-sets.
- Run `mysql-diag` and check the MySQL Server logs for errors.

---

**Connections per Second**

**/mysql/net/connections**

Connections per second made to the server.

Use: If the number of connections drastically changes or if apps are unable to connect, there might be a network or app issue.

Origin: Firehose
Envelope Type: Gauge
Unit: count
Frequency: 30 s (default)

**Recommended measurement**
(Average of all nodes / max connections), over last 1 minute

**Recommended alert thresholds**
Yellow warning: > 80%
Red critical: > 90%

**Recommended response**
- Run `mysql-diag` and check the MySQL Server logs for errors.
- When approaching 100% of max connections, Apps may be experiencing times when they cannot connect to the database. The connections per second for the cluster vary based on application instances and app utilization. If this threshold is met or exceeded
for an extended period of time, monitor app usage to ensure everything is behaving as expected.

Gorouter Metrics

Router File Descriptors

<table>
<thead>
<tr>
<th>Description</th>
<th>gorouter.file_descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The number of file descriptors currently used by the Gorouter job.</td>
</tr>
<tr>
<td>Use</td>
<td>Indicates an impending issue with the Gorouter. Without proper mitigation, it is possible for an unresponsive app to eventually exhaust available Gorouter file descriptors and cause route starvation for other apps running on PAS. Under heavy load, this unmitigated situation can also result in the Gorouter losing its connection to NATS and all routes being pruned. While a drop in gorouter.total_routes or an increase in gorouter.ms_since_last_registry_update helps to surface that the issue may already be occurring, alerting on gorouter.file_descriptors indicates that such an issue is impending.</td>
</tr>
<tr>
<td></td>
<td>The Gorouter limits the number of file descriptors to 100,000 per job. Once the limit is met, the Gorouter is unable to establish any new connections.</td>
</tr>
<tr>
<td></td>
<td>To reduce the risk of DDoS attacks, Pivotal recommends doing one or both of the following:</td>
</tr>
<tr>
<td></td>
<td>• Within PAS, set Max Connections Per Backend to define how many requests can be routed to any particular app instance. This prevents a single app from using all Gorouter connections. The value specified should be determined by the operator based on the use cases for that foundation. For example, Pivotal sets the number of connections to 500 for Pivotal Web Services.</td>
</tr>
<tr>
<td></td>
<td>• Add rate limiting at the load balancer level.</td>
</tr>
<tr>
<td>Origin</td>
<td>Firehose</td>
</tr>
<tr>
<td>Type</td>
<td>Gauge</td>
</tr>
<tr>
<td>Frequency</td>
<td>5 s</td>
</tr>
</tbody>
</table>

Recommended measurement: Maximum, per Gorouter job, over the last 5 minutes

Recommended alert thresholds:
- Yellow warning: 50,000 per job
- Red critical: 60,000 per job

Recommended response:
1. Identify which app(s) are requesting excessive connections and resolve the impacting issues with these apps.
2. If the above recommended mitigation steps have not already been taken, do so.
3. Consider adding more Gorouter VM resources to increase the number of available file descriptors.

Router Exhausted Connections

<table>
<thead>
<tr>
<th>Description</th>
<th>gorouter.backend_exhausted_conns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The lifetime number of requests that have been rejected by the Gorouter VM due to the Max Connections Per Backend limit being reached across all tried backends. The limit controls the number of concurrent TCP connections to any particular app instance and is configured within PAS.</td>
</tr>
<tr>
<td>Use</td>
<td>Indicates that PAS is mitigating risk to other applications by self-protecting the platform against one or more unresponsive applications. Increases in this metric indicate the need to investigate and resolve issues with potentially unresponsive applications. A rapid rate of change upward is concerning and should be assessed further.</td>
</tr>
</tbody>
</table>
Recommended measurement | Maximum delta per minute, per Gorouter job, over a 5-minute window
--- | ---
Recommended alert thresholds | Yellow warning: Dynamic
Red critical: Dynamic

Recommended response

1. If `gorouter.backend_exhausted_conns` spikes, first look to the Router Throughput metric `gorouter.total_requests` to determine if this measure is high or low in relation to normal bounds for this deployment.

2. If Router Throughput appears within normal bounds, it is likely that `gorouter.backend_exhausted_conns` is spiking due to an unresponsive application, possibly due to application code issues or underlying application dependency issues. To help determine the problematic application, look in access logs for repeated calls to one application. Then proceed to troubleshoot this application accordingly.

3. If Router Throughput also shows unusual spikes, the cause of the increase in `gorouter.backend_exhausted_conns` spikes is likely external to the platform. Unusual increases in load may be due to expected business events driving additional traffic to applications. Unexpected increases in load may indicate a DDoS attack risk.

### Router Throughput

**Description**

The lifetime number of requests completed by the Gorouter VM, emitted per Gorouter instance

Use: The aggregation of these values across all Gorouters provide insight into the overall traffic flow of a deployment. Unusually high spikes, if not known to be associated with an expected increase in demand, could indicate a DDoS risk. For performance and capacity management, consider this metric a measure of router throughput per job, converting it to requests-per-second, by looking at the delta value of `gorouter.total_requests` and deriving back to 1s, or `(gorouter.total_requests.delta)/5`, per Gorouter instance. This helps you see trends in the throughput rate that indicate a need to scale the Gorouter instances. Use the trends you observe to tune the threshold alerts for this metric.

**Origin:** Firehose

**Type:** Counter (Integer)

**Frequency:** 5 s

**Recommended measurement** | Average over the last 5 minutes of the derived per second calculation
--- | ---

**Recommended alert thresholds**

<table>
<thead>
<tr>
<th>Level</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow warning</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Red critical</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>

**Recommended response**

For optimizing the Gorouter, consider the requests-per-second derived metric in the context of router latency and Gorouter VM CPU utilization. From performance and load testing of the Gorouter, Pivotal has observed that at approximately 2500 requests per second, latency can begin to increase. To increase throughput and maintain low latency, scale the Gorouters either horizontally or vertically and watch that the `system.cpu.user` metric for the Gorouter stays in the suggested range of 60-70% CPU Utilization.

### Router Handling Latency

**gorouter.latency**

The time in milliseconds that the Gorouter takes to handle requests to backend endpoints, which include both applications routable platform system APIs like Cloud Controller and UAA. This is the average round trip response time, which includes router handling.
**Time Since Last Route Register Received**

<table>
<thead>
<tr>
<th>Description</th>
<th>gorouter.ms_since_last_registry_update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in milliseconds since the last route register was received, emitted per Gorouter instance.</td>
<td></td>
</tr>
<tr>
<td>Use: Indicates if routes are not being registered to apps correctly.</td>
<td></td>
</tr>
<tr>
<td>Origin: Firehose</td>
<td></td>
</tr>
<tr>
<td>Type: Gauge (Float in ms)</td>
<td></td>
</tr>
<tr>
<td>Frequency: 30 s</td>
<td></td>
</tr>
<tr>
<td>Recommended measurement</td>
<td>Maximum over the last 5 minutes</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
<td>Yellow warning: N/A</td>
</tr>
<tr>
<td>Red critical: &gt; 30,000 This threshold is suitable for normal platform usage. It alerts if it has been at least 30 seconds since the Gorouter last received a message from an app.</td>
<td></td>
</tr>
<tr>
<td>Recommended response</td>
<td>1. Search the Gorouter and Route Emitter logs for connection issues to NATS.</td>
</tr>
<tr>
<td></td>
<td>2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing.</td>
</tr>
<tr>
<td></td>
<td>3. Look more broadly at the health of all VMs, particularly Diego-related VMs.</td>
</tr>
<tr>
<td></td>
<td>4. If problems persist, pull the Gorouter and Route Emitter logs and contact Pivotal Support to say there are consistently long delays in route registry.</td>
</tr>
</tbody>
</table>

**Router Error: 502 Bad Gateway**

<table>
<thead>
<tr>
<th>Description</th>
<th>gorouter.bad_gateways</th>
</tr>
</thead>
<tbody>
<tr>
<td>The lifetime number of bad gateways, or 502 responses, from the Gorouter itself, emitted per Gorouter instance. The Gorouter emits a 502 bad gateway error when it has a route in the routing table and, in attempting to make a connection to the backend, finds that the backend does not exist.</td>
<td></td>
</tr>
<tr>
<td>Use: Indicates that route tables might be stale. Stale routing tables suggest an issue in the route register management plane, which indicates that something has likely changed with the locations of the containers.</td>
<td></td>
</tr>
</tbody>
</table>
Always investigate unexpected increases in this metric.

**Origin:** Firehose  
**Type:** Count (Integer, Lifetime)  
**Frequency:** 5 s

### Recommended measurement

Maximum delta per minute over a 5-minute window

### Recommended alert thresholds

**Yellow warning:** Dynamic  
**Red critical:** Dynamic

### Recommended response

1. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS.
2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing.
3. Look broadly at the health of all VMs, particularly Diego-related VMs.
4. If problems persist, pull Gorouter and Route Emitter logs and contact Pivotal Support to say there has been an unusual increase in Gorouter bad gateway responses.

---

**Router Error: Server Error**

<table>
<thead>
<tr>
<th>gorouter.responses.5xx</th>
</tr>
</thead>
</table>

**Description**

The lifetime number of requests completed by the Gorouter VM for HTTP status family 5xx, server errors, emitted per Gorouter instance.

Use: A repeatedly crashing app is often the cause of a big increase in 5xx responses. However, response issues from apps can also cause an increase in 5xx responses. Always investigate an unexpected increase in this metric.

**Origin:** Firehose  
**Type:** Counter (Integer)  
**Frequency:** 5 s

### Recommended measurement

Maximum delta per minute over a 5-minute window

### Recommended alert thresholds

**Yellow warning:** Dynamic  
**Red critical:** Dynamic

### Recommended response

1. Look for out-of-memory errors and other app-level errors.
2. As a temporary measure, ensure that the troublesome app is scaled to more than one instance.

---

**Number of Gorouter Routes Registered**

<table>
<thead>
<tr>
<th>gorouter.total_routes</th>
</tr>
</thead>
</table>

**Description**

The current total number of routes registered with the Gorouter, emitted per Gorouter instance.

Use: The aggregation of these values across all Gorouters indicates uptake and gives a picture of the overall growth of the environment for capacity planning.

Pivotal also recommends alerting on this metric if the number of routes falls outside of the normal range for your deployment. Dramatic decreases in this metric volume may indicate a problem with the route registration process, such as an app outage, or that something in the route register management plane has failed.

If visualizing these metrics on a dashboard, gorouter.total_routes can be helpful for visualizing dramatic drops. However, for alerting purposes, the gorouter.ms_since_last_registry_update metric is more valuable for quicker identification of Gorouter issues. Alerting thresholds for gorouter.total_routes should focus on dramatic increases or decreases out of expected range.

**Origin:** Firehose  
**Type:** Gauge (Float)
**Recommended measurement**  
5-minute average of the per second delta

**Recommended alert thresholds**  
- Yellow warning: Dynamic  
- Red critical: Dynamic

**Recommended response**
1. For capacity needs, scale up or down the Gorouter VMs as necessary.
2. For significant drops in current total routes, see the `gorouter.ms_since_last_registry_update` metric for additional context.
3. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS.
4. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing.
5. Look broadly at the health of all VMs, particularly Diego-related VMs.
6. If problems persist, pull the Gorouter and Route Emitter logs and contact Pivotal Support.

### Number of Route Registration Messages Sent and Received

This KPI is based on the following metrics:
- `route_emitter.HTTPRouteNATSMessagesEmitted` reports the lifetime number of route registration messages sent by the Route Emitter component. The metric is emitted for each Route Emitter.
- `gorouter.registry_message.route-emitter` reports the lifetime number of route registration messages received by the Gorouter. The metric is emitted for each Gorouter instance.

Dynamic configuration that enables the Gorouter to route HTTP requests to apps is published by the Route Emitter component colocated on each Diego cell to the NATS clustered message bus. All router instances subscribed to this message bus receive the same configuration. (Router instances within an isolation segment receive configuration only for cells in the same isolation segment.)

As Gorouters prune app instances from the route when a TTL expires, each Route Emitter periodically publishes the routing configuration for the app instances on the same cell.

Therefore, the aggregate number of route registration messages published by all the Route Emitters should be equal to the number of messages received by each Gorouter instance.

**Use:** A difference in the rate of change of these metrics is an indication of an issue in the control plane responsible for updating the routers with changes to the routing table.

Pivotal recommends alerting when the number of messages received per second for a given router instance falls below the sum of messages emitted per second across all Route Emitters.

If visualizing these metrics on a dashboard, look for increases in the difference between the rate of messages received and sent. If the number of messages received by a Gorouter instance drops below the sum of messages sent by the Route Emitters, this is an indication of a problem in the control plane.

**Origin:** Firehose  
**Type:** Counter  
**Frequency:** With each event

**Recommended measurement**  
Difference of 5-minute average of the per second deltas for `gorouter.registry_message.route-emitter` and sum of `route_emitter.HTTPRouteNATSMessagesEmitted` for all Route Emitters

**Recommended alert thresholds**  
- Yellow warning: Dynamic  
- Red critical: Dynamic

1. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS.
2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing.
3. Look broadly at the health of all VMs, particularly Diego-related VMs.

4. If problems persist, pull the Gorouter and Route Emitter logs and contact Pivotal Support.

**UAA Metrics**

**UAA Throughput**

<table>
<thead>
<tr>
<th><strong>uaa.requests.global.completed.count</strong></th>
</tr>
</thead>
</table>

**Description**

The lifetime number of requests completed by the UAA VM, emitted per UAA instance. This number includes health checks.

**Use:** For capacity planning purposes, the aggregation of these values across all UAA instances can provide insight into the overall load that UAA is processing. It is recommended to alert on unexpected spikes per UAA instance. Unusually high spikes, if they are not associated with an expected increase in demand, could indicate a DDoS risk and should be investigated.

For performance and capacity management, look at the UAA Throughput metric as either a requests-completed-per-second or requests-completed-per-minute rate to determine the throughput per UAA instance. This helps you see trends in the throughput rate that may indicate a need to scale UAA instances. Use the trends you observe to tune the threshold alerts for this metric.

From performance and load testing of UAA, Pivotal has observed that while UAA endpoints can have different throughput behavior, once throughput reaches its peak value per VM, it stays constant and latency increases.

**Origin:** Firehose  
**Type:** Gauge (Integer), emitted value increments over the lifetime of the VM like a counter  
**Frequency:** 5 s

**Recommended measurement**

Average over the last 5 minutes of the derived requests-per-second or requests-per-minute rate, per instance

**Recommended alert thresholds**

Yellow warning: Dynamic  
Red critical: Dynamic

**Recommended response**

For optimizing UAA, consider this metric in the context of UAA Request Latency and UAA VM CPU Utilization. To increase throughput and maintain low latency, scale the UAA VMs horizontally by editing the number of your UAA VM instances in the Resource Config pane of the PAS tile and ensure that the system.cpu.user metric for UAA is not sustained in the suggested range of 80-90% maximum CPU utilization.

**UAA Request Latency**

<table>
<thead>
<tr>
<th><strong>gorouter.latency.uaa</strong></th>
</tr>
</thead>
</table>

**Description**

Time in milliseconds that UAA took to process a request that the Gorouter sent to UAA endpoints.

**Use:** Indicates how responsive UAA has been to requests sent from the Gorouter. Some operations may take longer to process, such as creating bulk users and groups. It is important to correlate latency observed with the endpoint and evaluate this data in the context of overall historical latency from that endpoint. Unusual spikes in latency could indicate the need to scale UAA VMs.

This metric is emitted only for the routers serving the UAA system component and is not emitted per isolation segment even if you are using isolated routers.

**Origin:** Firehose  
**Type:** Gauge (Float in ms)  
**Frequency:** Emitted per Gorouter request to UAA

**Recommended measurement**

Maximum, per job, over the last 5 minutes
Recommended alert thresholds

Recommended response

Latency depends on the endpoint and operation being used. It is important to correlate the latency with the endpoint and evaluate this data in the context of the historical latency from that endpoint.

1. Inspect which endpoints requests are hitting. Use historical data to determine if the latency is unusual for that endpoint. A list of UAA endpoints is available in the UAA API documentation.

2. If it appears that UAA needs to be scaled due to ongoing traffic congestion, do not scale based on the latency metric alone. You should also ensure that the `system.cpu.user` metric for UAA stays in the suggested range of 80-90% maximum CPU utilization.

3. Resolve high utilization by scaling UAA VMs horizontally. To scale UAA, navigate to the Resource Config pane of the PAS tile and edit the number of your UAA VM instances.

UAA Requests In Flight

The number of requests UAA is currently processing (in-flight requests), emitted per UAA instance.

Use: Indicates how many concurrent requests are currently in flight for the UAA instance. Unusually high spikes, if they are not associated with an expected increase in demand, could indicate a DDoS risk.

From performance and load testing of the UAA component, Pivotal has observed that the number of concurrent requests impacts throughput and latency. The UAA Requests In Flight metric helps you see trends in the request rate that may indicate the need to scale UAA instances. Use the trends you observe to tune the threshold alerts for this metric.

Recommended alert thresholds

Recommended response

To increase throughput and maintain low latency when the number of in-flight requests is high, scale UAA VMs horizontally by editing the UAA VM field in the Resource Config pane of the PAS tile. Ensure that the `system.cpu.user` metric for UAA is not sustained in the suggested range of 80-90% maximum CPU utilization.

System (BOSH) Metrics

VM Health

The system is healthy, and 0 means the system is not healthy.

Use: This is the most important BOSH metric to monitor. It indicates if the VM emitting the metric is healthy. Review this metric for all VMs to estimate the overall health of the system.

Multiple unhealthy VMs signals problems with the underlying IAAS layer.

Recommended measurement

Average over the last 5 minutes
| **Recommended alert thresholds** | Yellow warning: N/A  
Red critical: < 1 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recommended response</strong></td>
<td>Investigate CF logs for the unhealthy component(s).</td>
</tr>
</tbody>
</table>

### VM Memory Used

**system.mem.percent**

| **Description** | System Memory — Percentage of memory used on the VM  
Use: Set an alert and investigate if the free RAM is low over an extended period. |
|-----------------|-------------------------------------------------------------------------------------------------|
| **Origin**      | Firehose  
**Type** | Gauge (%)  
**Frequency** | 60 s |
| **Recommended measurement** | Average over the last 10 minutes |
| **Recommended alert thresholds** | Yellow warning: ≥ 80%  
Red critical: ≥ 90% |
| **Recommended response** | The response depends on the job the metric is associated with. If appropriate, scale affected jobs out and monitor for improvement. |

### VM Disk Used

**system.disk.system.percent**

| **Description** | System disk — Percentage of the system disk used on the VM  
Use: Set an alert to indicate when the system disk is almost full. |
|-----------------|-------------------------------------------------------------------------------------------------|
| **Origin**      | Firehose  
**Type** | Gauge (%)  
**Frequency** | 60 s |
| **Recommended measurement** | Average over the last 30 minutes |
| **Recommended alert thresholds** | Yellow warning: ≥ 80%  
Red critical: ≥ 90% |
| **Recommended response** | Investigate what is filling the jobs system partition.  
This partition should not typically fill because BOSH deploys jobs to use ephemeral and persistent disks. |

### VM Ephemeral Disk Used

**system.disk.ephemeral.percent**

| **Description** | Ephemeral disk — Percentage of the ephemeral disk used on the VM  
Use: Set an alert and investigate if the ephemeral disk usage is too high for a job over an extended period. |
|-----------------|-------------------------------------------------------------------------------------------------|
| **Origin**      | Firehose  
**Type** | Gauge (%)  
**Frequency** | 60 s |
| **Recommended measurement** | Average over the last 30 minutes |
| **Recommended alert thresholds** | Yellow warning: ≥ 80%  
Red critical: ≥ 90% |

1. Run `bosh vms --details` to view jobs on affected deployments.
### VM Persistent Disk Used

<table>
<thead>
<tr>
<th>system.disk.persistent.percent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Persistent disk — Percentage of persistent disk used on the VM</td>
</tr>
<tr>
<td><strong>Use:</strong> Set an alert and investigate further if the persistent disk usage for a job is too high over an extended period.</td>
</tr>
<tr>
<td><strong>Origin:</strong> Firehose</td>
</tr>
<tr>
<td><strong>Type:</strong> Gauge (%)</td>
</tr>
<tr>
<td><strong>Frequency:</strong> 60 s</td>
</tr>
<tr>
<td><strong>Recommended measurement</strong></td>
</tr>
<tr>
<td>Average over the last 30 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
</tr>
<tr>
<td>Yellow warning: ≥ 80%</td>
</tr>
<tr>
<td>Red critical: ≥ 90%</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
</tr>
<tr>
<td>1. Run <code>bosh vms --details</code> to view jobs on affected deployments.</td>
</tr>
<tr>
<td>2. Determine cause of the data consumption, and, if appropriate, increase disk space or scale out affected jobs.</td>
</tr>
</tbody>
</table>

### VM CPU Utilization

<table>
<thead>
<tr>
<th>system.cpu.user</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>CPU utilization — The percentage of CPU spent in user processes</td>
</tr>
<tr>
<td><strong>Use:</strong> Set an alert and investigate further if the CPU utilization is too high for a job.</td>
</tr>
<tr>
<td>For monitoring Gorouter performance, CPU utilization of the Gorouter VM is the recommended key capacity scaling indicator. For more information, see <a href="https://www.pivotal.io">Gorouter Latency and Throughput</a>.</td>
</tr>
<tr>
<td><strong>Origin:</strong> Firehose</td>
</tr>
<tr>
<td><strong>Type:</strong> Gauge (%)</td>
</tr>
<tr>
<td><strong>Frequency:</strong> 60 s</td>
</tr>
<tr>
<td><strong>Recommended measurement</strong></td>
</tr>
<tr>
<td>Average over the last 5 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
</tr>
<tr>
<td>Yellow warning: ≥ 85%</td>
</tr>
<tr>
<td>Red critical: ≥ 95%</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
</tr>
<tr>
<td>1. Investigate the cause of the spike.</td>
</tr>
<tr>
<td>2. If the cause is a normal workload increase, then scale up the affected jobs.</td>
</tr>
</tbody>
</table>
Key Capacity Scaling Indicators

This topic describes key capacity scaling indicators that operators monitor to determine when they need to scale their Pivotal Application Service (PAS) deployments.

Pivotal provides these indicators to operators as general guidance for capacity scaling. Each indicator is based on platform metrics from different components. This guidance is applicable to most PAS v2.4 deployments. Pivotal recommends that operators fine-tune the suggested alert thresholds by observing historical trends for their deployments.

Diego Cell Capacity Scaling Indicators

There are three key capacity scaling indicators recommended for Diego cell:

- **Diego Cell Memory Capacity** is a measure of the percentage of remaining memory capacity
- **Diego Cell Disk Capacity** is a measure of the percentage of remaining disk capacity
- **Diego Cell Container Capacity** is a measure of the percentage of remaining container capacity

### Diego Cell Memory Capacity

<table>
<thead>
<tr>
<th>Description</th>
<th>Percentage of remaining memory capacity for a given cell. Monitor this derived metric across all cells in a deployment.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The metric <code>rep.CapacityRemainingMemory</code> indicates the remaining amount in MiB of memory available for this cell to allocate to containers.</td>
</tr>
<tr>
<td></td>
<td>The metric <code>rep.CapacityTotalMemory</code> indicates the total amount in MiB of memory available for this cell to allocate to containers.</td>
</tr>
<tr>
<td>Purpose</td>
<td>A best practice deployment of Cloud Foundry includes three availability zones (AZs). For these types of deployments, Pivotal recommends that you have enough capacity to suffer failure of an entire AZ.</td>
</tr>
<tr>
<td></td>
<td>The <strong>Recommended threshold</strong> assumes a three-AZ configuration. Adjust the threshold percentage if you have more or fewer AZs.</td>
</tr>
<tr>
<td>Recommended thresholds</td>
<td>&lt; avg(35%)</td>
</tr>
<tr>
<td>How to scale</td>
<td>Scale up your Diego Cells</td>
</tr>
</tbody>
</table>
| Additional details | **Origin**: Firehose  
**Type**: Gauge (%)  
**Frequency**: Emitted every 60 s  
**Applies to**: cf.diego_cells |
| Alternative Metric | PCF Healthwatch expresses this indicator with the metric `healthwatch.Diego.TotalPercentageAvailableMemoryCapacity.5M` |

### Diego Cell Disk Capacity

<table>
<thead>
<tr>
<th>Description</th>
<th>Percentage of remaining disk capacity for a given cell. Monitor this derived metric across all cells in a deployment.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The metric <code>rep.CapacityRemainingDisk</code> indicates the remaining amount in MiB of disk available for this cell to allocate to containers.</td>
</tr>
<tr>
<td></td>
<td>The metric <code>rep.CapacityTotalDisk</code> indicates the total amount in MiB of disk available for this cell to allocate to containers.</td>
</tr>
<tr>
<td>Purpose</td>
<td>A best practice deployment of Cloud Foundry includes three availability zones (AZs). For these types of deployments, Pivotal recommends that you have enough capacity to suffer failure of an entire AZ.</td>
</tr>
</tbody>
</table>
Purpose
The **Recommended threshold** assumes a three-AZ configuration. Adjust the threshold percentage if you have more or fewer AZs.

Recommended thresholds
< avg(35%)

How to scale
Scale up your Diego Cells

Additional details
- **Origin:** Firehose
- **Type:** Gauge (%)
- **Frequency:** Emitted every 60 s
- **Applies to:** cf:diego_cells

Alternative Metric
PCF Healthwatch expresses this indicator with the metric
healthwatch.Diego.TotalPercentageAvailableDiskCapacity.5M

Diego Cell Container Capacity

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage of remaining container capacity for a given cell. Monitor this derived metric across all cells in a deployment.</td>
</tr>
<tr>
<td>The metric <code>rep.CapacityRemainingContainers</code> indicates the remaining number of containers this cell can host.</td>
</tr>
<tr>
<td>The metric <code>rep.CapacityTotalContainers</code> indicates the total number of containers this cell can host.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>A best practice deployment of Cloud Foundry includes three availability zones (AZs). For these types of deployments, Pivotal recommends that you have enough capacity to suffer failure of an entire AZ.</td>
</tr>
<tr>
<td>The <strong>Recommended threshold</strong> assumes a three-AZ configuration. Adjust the threshold percentage if you have more or fewer AZs.</td>
</tr>
</tbody>
</table>

Recommended thresholds
< avg(35%)

How to scale
Scale up your Diego Cells

Additional details
- **Origin:** Firehose
- **Type:** Gauge (%)
- **Frequency:** Emitted every 60 s
- **Applies to:** cf:diego_cells

Alternative Metric
PCF Healthwatch expresses this indicator with the metric
healthwatch.Diego.TotalPercentageAvailableContainerCapacity.5M

Firehose Performance Scaling Indicators
Pivotal recommends two key capacity scaling indicators for monitoring Firehose performance.

Log Transport Loss Rate

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This derived value represents the loss rate occurring as messages are transported from the Loggregator Agent components through the Doppler components to the Firehose endpoints.</td>
</tr>
<tr>
<td>Metric <code>loggregator.doppler.ingress</code> represents the number of messages entering Dopplers for transport through the firehose, and <code>loggregator.doppler.dropped</code> represents the number of messages dropped without delivery.</td>
</tr>
<tr>
<td>Messages include the combined stream of logs from all apps and the metrics data from Cloud Foundry components.</td>
</tr>
<tr>
<td>For more information about Loggregator components, see <a href="#">Loggregator Architecture</a>.</td>
</tr>
</tbody>
</table>

Excessive dropped messages can indicate the Dopplers or Traffic Controllers are not processing messages quickly enough.
The recommended scaling indicator is a dropped message rate greater than \([0.01]\). This scaling indicator is calculated by expressing the total number of dropped messages as a percentage of the total throughput and scale.

Doppler emits two separate dropped metrics, one for ingress and one for egress. The envelopes have a \([\text{direction}]\) tag with a value of \([\text{ingress}]\). For this indicator, use the metric with a \([\text{direction}]\) tag with a value of \([\text{ingress}]\).

Recommended thresholds:

<table>
<thead>
<tr>
<th>Scale indicator</th>
<th>≥ 0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>If alerting:</td>
<td></td>
</tr>
<tr>
<td>Yellow warning:</td>
<td>≥ 0.005</td>
</tr>
<tr>
<td>Red critical:</td>
<td>≥ 0.01</td>
</tr>
</tbody>
</table>

How to scale:

Scale up the number of Traffic Controller and Doppler instances.

Note: At approximately 40 Doppler instances and 20 Traffic Controller instances, horizontal scaling is no longer useful for improving Firehose performance. To improve performance, add vertical scale to the existing Doppler and Traffic Controller instances by increasing CPU resources.

Additional details:

Origin: Firehose
Type: Gauge (float)
Frequency: Base metrics are emitted every 5 s
Applies to: cf:doppler

Alternative Metric:

PCF Healthwatch expresses this indicator with the metrics \([\text{healthwatch.Firehose.LossRate.1H}]\) and \([\text{healthwatch.Firehose.LossRate.1M}]\).

Doppler Message Rate Capacity

\[
\frac{\text{loggregator.doppler.ingress (sum across instances)}}{\text{current number of Doppler instances}}
\]

Description:

This derived value represents the average rate of envelopes (messages) per Doppler instance. Deriving this into a per-Doppler envelopes-per-second, or envelopes-per-minute, rate can indicate the need to scale when Doppler instances are at their recommended maximum load.

Purpose:

The recommended scaling indicator is to look at the average load on the Doppler instances, and increase the number of Doppler instances when the derived rate is 16,000 envelopes per second, or 1 million envelopes per minute.

Recommended thresholds:

Scale indicator: ≥ 16,000 envelopes per second (or 1 million envelopes per minute)

How to scale:

Increase the number of Doppler VMs in the Resource Config pane of the PAS tile.

Additional details:

Origin: Firehose
Type: Gauge (float)
Frequency: Emitted every 15 s
Applies to: cf:doppler

Alternative Metric:

PCF Healthwatch expresses this indicator with the metric \([\text{healthwatch.Doppler.MessagesAverage.1M}]\).

Reverse Log Proxy Loss Rate

\[
\frac{\text{loggregator.rlp.dropped}}{\text{loggregator.rlp.ingress}}
\]

Description:

The loss rate of the reverse log proxies (RLP), that is, the total messages dropped as a percentage of the total traffic coming through the reverse log proxy. Total messages include only logs for bound applications.

This loss rate is specific to the RLP and does not impact the Firehose loss rate. For example, you can suffer lossiness in the RLP while not suffering any lossiness in the Firehose.

Purpose:

Excessive dropped messages can indicate that the RLP is overloaded and that the Traffic Controllers need to be scaled.

The recommended scaling indicator is to look at the maximum per minute loss rate over a 5-minute window and scale if the derived loss rate value grows greater than \([0.1]\).

Recommended thresholds:

Scale indicator: ≥ 0.1
If alerting:
Yellow warning: ≥ 0.01
Red critical: ≥ 0.1

How to scale:

Scale up the number of traffic controller instances to further balance log load.

Origin: Firehose
Firehose Consumer Scaling Indicator

Pivotal recommends the following scaling indicator for monitoring the performance of consumers of the Firehose.

Slow Consumer Drops

doppler_proxy.slow_consumer

| Description | Within PAS, metrics and logs enter the Firehose for transport and exit out of the platform via a consumer nozzle. If the consuming downstream system fails to keep up with the exiting stream of metrics, the Firehose is forced to close the connection to protect itself from back-pressure. The Firehose increments slow_consumer with each connection that it closes because a consumer could not keep up. |
| Purpose | This metric indicates that a Firehose consumer, such as a monitoring tool nozzle, is ingesting too slowly. If this number is anomalous, it may result in the downstream monitoring tool not having all expected data, even though that data was successfully transported through the Firehose. This can also be confirmed via the Loggregator emitted log message TB: Output channel too full. Dropped N messages, where N is the number of dropped messages. |
| Recommended thresholds | Scale indicator: It is recommended to scale when the rate of Firehose Slow Consumer Drops is anomalous for a given environment. |
| How to scale | Scale up the number of nozzle instances. You can scale a nozzle using the subscription ID specified when the nozzle connects to the Firehose. If you use the same subscription ID on each nozzle instance, the Firehose evenly distributes data across all instances of the nozzle. For example, if you have two nozzle instances with the same subscription ID, the Firehose sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the Firehose sends one-third of the data to each instance. If you want to scale a nozzle, the number of nozzle instances should match the number of Traffic Controller instances. |
| Additional details | Origin: Firehose Type: Counter Frequency: Emitted every 5 s Applies to: cf:doppler |

Reverse Log Proxy Egress Dropped Messages

rlp.dropped, direction: egress

| Description | Within PAS, logs and metrics enter Loggregator for transport and then egress through the Reverse Log Proxy (RLP). The RLP drops messages when consumers of the RLP, such as monitoring tool nozzles, ingest the exiting stream of logs and metrics too slowly. Note: The rlp.dropped metric includes both ingress and egress directions. To differentiate between ingress and egress, refer to the direction tag on the metric. |
| Purpose | This metric indicates that a consumer of logs and metrics from the RLP, such as a monitoring tool nozzle, is ingesting RLP messages too slowly. |
| Recommended thresholds | Scale indicator: Scale when the rate of rlp.dropped, direction: egress metrics is continuously increasing. |
| How to scale | Scale up the number of nozzle instances. The number of nozzle instances should match the number of Traffic Controller instances. You can scale a nozzle using the subscription ID specified when the nozzle connects to the RLP. If you use the same subscription ID on each nozzle instance, the RLP evenly distributes data across all instances of the nozzle. For example, if you have two nozzle instances with the same subscription ID, the RLP sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the RLP sends one-third of the data to each instance. |
| Additional details | Origin: Reverse Log Proxy Type: Counter Frequency: Emitted every 5 s |
CF Syslog Drain Performance Scaling Indicators

There are three key capacity scaling indicators recommended for CF Syslog Drain performance.

**Note:** These CF Syslog Drain scaling indicators are only relevant if your deployment contains apps using the CF syslog drain binding feature.

### Adapter Loss Rate

<table>
<thead>
<tr>
<th>cf-syslog-drain.adapter.dropped / cf-syslog-drain.adapter.ingress</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
</tr>
</tbody>
</table>
| **Recommended thresholds** | Scale indicator: ≥ 0.1
If alerting: Yellow warning: ≥ 0.01
Red critical: ≥ 0.1 |
| **How to scale** | Performance test your syslog server, review the logs of the syslog consuming system for intake and other performance issues that indicate a need to scale the consuming system. |
| **Additional details** | Origin: Firehose
Type: Counter (Integer)
Frequency: Emitted every 60 s
Applies to: cf-cf-syslog |
| **Alternative Metric** | PCF Healthwatch expresses this indicator with the metric: healthwatch.SyslogDrain.Adapter.LossRate.1M. |

### CF Syslog Drain Bindings Count

<table>
<thead>
<tr>
<th>cf-syslog-drain.drain_adapter.drain_bindings / number of Syslog Adapters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Recommended thresholds</strong></td>
</tr>
<tr>
<td><strong>How to scale</strong></td>
</tr>
</tbody>
</table>
| **Additional details** | Origin: Firehose
Type: Gauge (float)
Frequency: Emitted every 60 s
Applies to: cf-cf-syslog |
| **Alternative Metric** | PCF Healthwatch expresses this indicator with the metric: healthwatch.SyslogDrain.Adapter.BindingsAverage.5M. |

### Log Cache Scaling Indicator

Note: These CF Syslog Drain scaling indicators are only relevant if your deployment contains apps using the CF syslog drain binding feature.
Pivotal recommends the following scaling indicator for monitoring the performance of Log Cache.

**Log Cache Caching Duration**

<table>
<thead>
<tr>
<th>Description</th>
<th>log_cache.cache_period</th>
</tr>
</thead>
<tbody>
<tr>
<td>This metric indicates the age in milliseconds of the oldest data point stored in Log Cache.</td>
<td></td>
</tr>
</tbody>
</table>

**Purpose**

Log Cache stores all messages that are passed through the Firehose in an ephemeral in-memory store. The size of this store and the cache duration are dependent on the amount of memory available on the VM on which Log Cache runs. Some features of PAS rely on data being available in Log Cache, such as App Autoscaler.

Pivotal recommends scaling the VM on which Log Cache runs, so Log Cache can hold all messages that pass through Loggregator in the last 15 minutes, or 900000 milliseconds.

**Recommended thresholds**

*Scale indicator:* Scale the VM on which Log Cache runs when the cache period drops below 15 minutes, or 900000 milliseconds. Typically, Log Cache runs on the Doppler VM.

**How to scale**

Scale up the number of Doppler VMs or chose a VM type for Doppler that provides more memory.

**Additional details**

*Origin:* log-cache  
*Type:* Gauge  
*Frequency:* Emitted every 15 s  
*Applies to:* cf:log-cache

---

**Router Performance Scaling Indicator**

There is one key capacity scaling indicator recommended for Router performance.

**Router VM CPU Utilization**

<table>
<thead>
<tr>
<th>Description</th>
<th>system.cpu.user of the Gorouter VM(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU utilization of the Gorouter VM(s)</td>
<td></td>
</tr>
</tbody>
</table>

**Purpose**

High CPU utilization of the Gorouter VMs can increase latency and cause throughput, or requests per/second, to level-off. Pivotal recommends keeping the CPU utilization within a maximum range of 60-70% for best Gorouter performance.

If you want to increase throughput capabilities while also keeping latency low, Pivotal recommends scaling the Gorouter while continuing to ensure that CPU utilization does not exceed the maximum recommended range.

**Recommended thresholds**

*Scale indicator:* ≥ 60%  
*Yellow warning:* ≥ 60%  
*Red critical:* ≥ 70%

**How to scale**

Resolve high utilization by scaling the Gorouters horizontally or vertically by editing the Router VM in the Resource Config pane of the PAS tile.

**Additional details**

*Origin:* Firehose  
*Type:* Gauge (float)  
*Frequency:* Emitted every 60 s  
*Applies to:* cf:router

---

**UAA Performance Scaling Indicator**

There is one key capacity scaling indicator recommended for UAA performance.

**UAA VM CPU Utilization**

<table>
<thead>
<tr>
<th>Description</th>
<th>system.cpu.user of the UAA VM(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU utilization of the UAA VM(s)</td>
<td></td>
</tr>
</tbody>
</table>
Purpose

High CPU utilization of the UAA VMs can increase latency and cause throughput, or requests per/second, to level-off. Pivotal recommends keeping the CPU utilization within a maximum range of 80-90% for best UAA performance.

If you want to increase throughput capabilities while keeping latency low, Pivotal recommends scaling the UAA VMs and ensuring that CPU utilization does not exceed the maximum recommended range.

Recommended thresholds

<table>
<thead>
<tr>
<th>Scale indicator</th>
<th>≥ 80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>If alerting:</td>
<td></td>
</tr>
<tr>
<td>Yellow warning</td>
<td>≥ 80%</td>
</tr>
<tr>
<td>Red critical</td>
<td>≥ 90%</td>
</tr>
</tbody>
</table>

How to scale

Resolve high utilization by scaling UAA horizontally or vertically. To scale UAA, navigate to the Resource Config pane of the PAS tile and edit the number of your UAA VM instances or change the VM type to a type that utilizes more CPU cores.

Additional details

Origin: Firehose
Type: Gauge (float)
Frequency: Emitted every 60 s
Applies to: cf:uaa

NFS/WebDAV Backed Blobstore

There is one key capacity scaling indicator for external S3 external storage.

Note: This metric is only relevant if your deployment does not use an external S3 repository for external storage with no capacity constraints.

<table>
<thead>
<tr>
<th>system.disk.persistent.percent of NFS server VM(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>If applicable: Monitor the percentage of persistent disk used on the VM for the NFS Server job.</td>
</tr>
</tbody>
</table>

Purpose

If you do not use an external S3 repository for external storage with no capacity constraints, you must monitor the PAS object store to push new app and buildpacks.

If you use an internal NFS/WebDAV backed blobstore, consider scaling the persistent disk when it reaches 75% capacity.

Recommended thresholds

| ≥ 75% |

How to scale

Give your NFS Server additional persistent disk resources.

Additional details

Origin: Firehose
Type: Gauge (%)
Applies to: cf:nfs_server
Selecting and Configuring a Monitoring System

This topic describes considerations for selecting and configuring a system to continuously monitor Pivotal Cloud Foundry (PCF) component performance and health.

Selecting a Monitoring Platform

Pivotal recommends using PCF Healthwatch to monitor your deployment. PCF Healthwatch is a service tile developed and supported by Pivotal and available on Pivotal Network.

Many third-party systems can also be used to monitor a PCF deployment.

Monitoring Platform Types

Monitoring platforms support two types of monitoring:

- A dashboard for active monitoring when you are at a keyboard and screen
- Automated alerts for when your attention is elsewhere

Some monitoring solutions offer both in one package. Others require putting the two pieces together.

Monitoring Platforms

There are many monitoring options available, both open source and commercial products. Some commonly-used platforms among PCF customers include:

- PCF Healthwatch by Pivotal
- PCF Partner Services available on Pivotal Network:
  - AppDynamics
  - Datadog
  - Dynatrace
  - New Relic
  - SignalFx
  - WaveFront by VMware
- Other Commercial Services
  - VMware vRealize Operations (vROPS)
- Open Source Tooling
  - Prometheus + Grafana
  - OpenTSDB

Pivotal Cloud Ops Tools

The Pivotal Cloud Ops Team manages two types of deployments for internal Pivotal use: open-source Cloud Foundry, and Pivotal Cloud Foundry (PCF).

For Cloud Foundry, Pivotal Cloud Ops uses several monitoring tools. The Datadog Config repository provides an example of how the Pivotal Cloud Ops team uses a customized Datadog dashboard to monitor the health of its open-source Cloud Foundry deployments.

To monitor its PCF deployments, Pivotal Cloud Ops leverages a combination of PCF Healthwatch and Google Stackdriver.

Key Inputs for Platform Monitoring
BOSH VM and PCF Component Health Metrics

Most monitoring service tiles for PCF come packaged with the Firehose nozzle necessary to extract the BOSH and CF metrics leveraged for platform monitoring. Nozzles are programs that consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and to forward it to other apps and services.

The nozzles gather the component logs and metrics streaming from the Loggregator Firehose endpoint. For more information about the Firehose, see Loggregator Architecture.

As of PCF v2.0, both BOSH VM Health metrics and Cloud Foundry component metrics stream through the Firehose by default.

- PCF component metrics originate from the Metron agents on their source components, then travel through Dopplers to the Traffic Controller.
- The Traffic Controller aggregates both metrics and log messages system-wide from all Dopplers, and emits them from its Firehose endpoint.

The following topic lists high-signal-value metrics and capacity scaling indicators in a PCF deployment:

- Key Performance Indicators
- Key Capacity Scaling Indicators

Continuous Functional Smoke Tests

PCF includes smoke tests, which are functional unit and integration tests on all major system components. By default, whenever an operator upgrades to a new version of PAS, these smoke tests run as a post-deploy errand.

Pivotal recommends additional higher-resolution monitoring by the execution of continuous smoke tests, or Service Level Indicator tests, that measure user-defined features and check them against expected levels.

- PCF Healthwatch automatically executes these tests for PAS Service Level Indicators.
- The Pivotal Cloud Ops CF Smoke Tests repository offers additional testing examples.

See the Metrics topic in the Concourse documentation for how to set up Concourse to generate custom component metrics.

Warning and Critical Thresholds

To properly configure your monitoring dashboard and alerts, you must establish what thresholds should drive alerting and red/yellow/green dashboard behavior.

Some key metrics have more fixed thresholds, with similar threshold numbers recommended across different foundations and use cases. These metrics tend to revolve around the health and performance of key components that can impact the performance of the entire system.

Other metrics of operational value are more dynamic in nature. This means that you must establish a baseline and yellow/red thresholds suitable for your system and its use cases. You can establish initial baselines by watching values of key metrics over time and noting what seems to be a good starting threshold level that divides acceptable and unacceptable system performance and health.

Continuous Evolution

Effective platform monitoring requires continuous evolution.

After you establish initial baselines, Pivotal recommends that you continue to refine your metrics and tests to maintain the appropriate balance between early detection and reducing unnecessary alert fatigue. The dynamic measures recommended in Key Performance Indicators and Key Capacity Scaling Indicators should be revisited on occasion to ensure they are still appropriate to the current system configuration and its usage patterns.
Overview

Pivotal Application Service (PAS) for Windows enables operators to provision, operate, and manage Windows cells on Pivotal Cloud Foundry (PCF).

After operators install the PAS for Windows tile on the Ops Manager Installation Dashboard, developers can push .NET applications to Windows cells using the Cloud Foundry Command Line Interface (cf CLI).

For more information about how PAS for Windows works, see the Product Architecture topic.

For information about deploying .NET apps to PAS for Windows, see the Tips for .NET Developers topic.

For the PAS for Windows release notes, see the PAS for Windows v2.5 Release Notes topic.

Requirements

To install the PAS for Windows tile, you must have the following:

- Ops Manager v2.5 and PAS v2.5 deployed to vSphere, Amazon Web Services (AWS), Google Cloud Platform (GCP), or Azure
- A Windows stemcell, which you can obtain by following the directions in Downloading or Creating Windows Stemcells.

The minimum resource requirements for each Windows cell are as follows:

- Disk size: 64 GB (100 GB recommended)
- Memory: 16 GB
- CPUs: 4

Contents

About PAS for Windows

- Product Architecture

Installation Guide

- Downloading or Creating Windows Stemcells
- Creating a vSphere Windows Stemcell
- Installing and Configuring PASW
- Windows Cells in Isolation Segments
- Migrating Apps to PAS for Windows

Admin Guide

- Upgrading Windows Cells
- Troubleshooting Windows Cells

Developer Guide
Limitations

PAS for Windows has the following limitations:

- Developers cannot push Docker or other OCI-compatible images to Windows Diego cells.
- Container-to-container networking is not yet available for Windows-hosted applications.
- Volume Services are not yet available for Windows-hosted applications.
- OpenStack is currently not supported for PASW. Contact your Pivotal representative for information on OpenStack deployments.
- Due to a known issue in Windows Server OS, applications hosted on PAS for Windows cannot route traffic when deployed with the IPsec add-on for PCF.
Deploying .NET Apps

This topic lists resources for .NET app developers.

For general information about app deployment, see Deploy an Application.

.NET Core on Linux

.NET Core Buildpack describes how to push .NET Core apps to Linux cells on Cloud Foundry using the .NET Core buildpack. For supported ASP.NET Core versions, see .NET Core buildpack release notes in GitHub.

.NET Framework on Windows

HWC Buildpack describes how to push .NET Framework apps to Pivotal Application Service for Windows (PASW) using the HWC buildpack. It also describes what features of IIS (Internet Information Services) and HWC (Hostable Web Core) are supported, and how to configure your .NET Framework apps to run on PASW.

Console Applications in the .NET Cookbook provides information about pushing compiled .NET Framework Console apps to PASW using the binary buildpack.

.NET Core on Windows

Binary Buildpack describes how to push compiled .NET Core apps to Windows cells on PASW using the binary buildpack.

Other Resources

Tips for .NET Framework Developers describes common errors and challenges faced when pushing .NET Framework apps to PASW.

The .Net Cookbook provides code samples, guidelines, tips, and other recipes for making changes to both new and existing .NET Framework and .NET Core apps to run successfully on PASW.
Product Architecture

This topic describes the architecture of Windows cells that PAS for Windows deploys to run containerized .NET apps, and the stemcells that it supplies to BOSH as the operating system for the Windows cell VMs.

Overview

Operators who want to run Windows cells in PCF to enable developers to push .NET apps can deploy the PAS for Windows tile.

Deploying this tile creates a separate BOSH deployment populated with the Garden Windows release, which runs on a Windows cell built from a Windows stemcell.

Once the Windows cell is running, developers can specify a Windows stack when pushing .NET apps from the command line. PCF passes the app to the Windows cell in the PAS for Windows BOSH deployment. The diagram below illustrates the process.

About Windows Cells

App instances in PCF run inside containers. Garden is the API that creates and manages these containers. An implementation equivalent to that on Linux cells provides this infrastructure on Windows cells, utilizing native Windows Server Containers.

By installing the PAS for Windows tile, operators create a Windows cell from a stemcell that contains the Windows Server operating system. Garden on Windows uses Windows Containers to isolate resources on Windows cells that Cloud Foundry manages alongside Linux cells.

Components

A Windows cell includes the following components:

- **Guardian**: Implements the Garden API on Windows
- **Metron Agent**: Forwards app logs, errors, and metrics to the Loggregator system
- **BOSH Agent**: Executes instructions from the BOSH Director
- **Consul Client**: Registers the cell as a service in a Consul cluster
- **Diego Rep**: Runs and manages Tasks and Long Running Processes

The following diagram illustrates the architecture of a Windows cell:
Container Implementation

Garden on Windows uses the following runtime plugins to create and manage Windows Containers for PAS:

- Container plugin `winc`: Creates OCI-compliant containers, executes processes in the containers, and sets their CPU and RAM limits.
- Network plugin `winc-network`: Creates a network compartment for the container, applies its DNS settings, and defines its inbound/outbound network access rules.
- Rootfs image plugin `groot`: Sets up the container filesystem volume and uses the FSRM API to define its disk usage quotas.

About Windows Stemcells

A "stemcell" is a customized operating system image containing the filesystem for BOSH-managed virtual machines. When deployed, the operating system includes the BOSH Agent process, which is dedicated to communicating with the orchestrating VM, the BOSH Director. The BOSH Agent executes and monitors BOSH jobs on its VM.


See Downloading or Creating Windows Stemcells for documentation about how to obtain or create a stemcell for PAS for Windows.
Downloading or Creating Windows Stemcells

This topic describes how to download or create the stemcell that Pivotal Application Service (PAS) for Windows needs to create VMs on an infrastructure.

Windows Stemcells

BOSH needs a stemcell to supply the basic operating system for any Windows VM it creates. Depending on your IaaS, you can create or download a stemcell as follows:

- **Azure**: [See instructions below](#) on how to activate and download the Azure Windows Stemcell.
- **Google Cloud Platform (GCP)**: Download the [GCP light Stemcell for Windows 2019 Server](#) from the [Stemcells for PCF (Windows Server)](#) section of Pivotal Network.
- **vSphere**: Create a stemcell by following the directions in [Creating a vSphere Windows Stemcell](#).
- **Amazon Web Services (AWS)**: Download the [AWS light Stemcell for Windows 2019 Server](#) from the [Stemcells for PCF (Windows Server)](#) section of Pivotal Network.

Configuring the Azure Light Stemcell

On Azure, the stemcell exists as an offering in the Azure Marketplace. To start using the Azure light stemcell, perform the steps in the following sections:

- [Enable the Azure Light Stemcell](#)
- [Download the Azure Light Stemcell](#)

Enable the Azure Light Stemcell

To use the Azure light stemcell, you must first accept the corresponding Microsoft license agreement and enable the stemcell for your non-trial Azure Subscription through the Azure Marketplace as follows:

1. Navigate to [https://portal.azure.com](https://portal.azure.com) and log in.
2. From the options on the left side of the page, click **Create a resource**.
3. In the **Search the Marketplace** bar, search for **BOSH Stemcell for Windows Server 2019**.
5. Below the description, at the bottom of the page, click the blue banner that reads **Want to deploy programmatically? Get started ➔**.
6. Review the Terms of Use in the page “Configure Programmatic Deployment” that appears.
7. Under **Choose the subscriptions**, click **Enable** for each Azure subscription with which you want to use the stemcell.
8. Click **Save**.

Download the Azure Light Stemcell

To download the Azure light stemcell for use in Ops Manager, perform the following steps:

1. Navigate to [Pivotal Network](#).
2. Select **Azure Light Stemcell for Windows 2019 Server** from the **Release Download Files** section of the Stemcells for PCF (Windows) page.

For information about how to deploy and configure the PAS for Windows tile, see [Installing and Configuring PAS for Windows](#).
Creating a vSphere Windows Stemcell

This topic describes how to create the stemcell that Pivotal Application Service for Windows (PASW) needs to create VMs on vSphere.

**Note:** The instructions in this topic are based on vSphere 6.0 using vSphere Web Client.

### Overview

To create a Windows stemcell for vSphere, you create a base Windows VM from a volume-licensed ISO and subsequently maintain that base template with all Windows recommended security updates, but without the BOSH dependencies.

**Note:** The stemcell you create in this topic is based on Windows Server 2019.

The VM with security updates serves as the base for all future stemcells, produced from clones of that base VM. This enables you to build new stemcells without having to run Windows Updates from scratch each time. You can also use a “snapshot” feature to maintain an updated Windows image that does not contain the BOSH dependencies.

Pivotal recommends installing any available critical updates, and then rebuilding the stemcell from a clone of the original VM.

### Prerequisites

Before you create a vSphere Windows stemcell, you must have the following:

- A Windows Server 2019 ISO, from Microsoft Developer Network (MSDN) or Volume Licensing Service Center (VLSC). You can use an evaluation copy for testing, but Pivotal does not recommend an evaluation copy for production, as the licensing expires.

  **Note:** Pivotal recommends maintaining a separate, updated Windows VM based on this ISO to serve as the basis for the installation steps below. This enables you to apply Windows Updates and create new stemcells without having to reinstall all updates from scratch.

- A vSphere/vCenter account granted sufficient permissions to perform all of the following tasks:
  - Create a VM.
  - Configure a VM.
  - Open a VM in VM Remote Console on a local desktop.
  - Export a VM.
  - The ability to download/transfer files and software to a vCenter Windows VM.

### Files on Local Machine

As part of completing the procedures in this topic, you download the following files to your local machine:

- The latest release of stembuild.
- ovftool.

### Files on Windows VM

As part of completing the procedures in this topic, you download the following files to your Windows VM:

- `lgpo.exe` from the Microsoft Security Toolkit.
- `OpenSSH v7.7.2.0p1`.
- The BOSH PS Modules and BOSH Agent for the 2019 stemcell version you want to build.

**Note:** You must choose a stemcell version to build. Stemcells are versioned as MAJOR.MINOR, such as 2019.2. For more information about
Step 1: Create Base VM for Stemcell

This section describes how to create, configure, and verify a base Windows VM from a volume-licensed ISO.

Upload the Windows ISO

Perform the following steps to upload the Windows ISO:

1. Log in to vCenter.
2. Click Storage in the vCenter menu.
3. Choose a datastore and click or create the directory where you want the Windows ISO.
4. Click Upload a file to datastore, and upload the Windows ISO.

Create and Customize a New VM

Perform the following steps to create and customize a new VM:

1. In the vSphere client, click the VMs and Templates view to display the inventory objects.
2. Right-click an object and select New Virtual Machine > New Virtual Machine....
3. On the Select a creation type page, select Create a new virtual machine and click Next.

4. On the Select a name and folder page, perform the following steps:
   a. Enter a name for the VM.
   b. Select a location for the VM.
   c. Click Next.

Note: You might need to install the vSphere client web plugin to upload through your browser, or scp the file directly to the datastore server. For more information, see the VMware vSphere documentation.
5. On the **Select a compute resource** page, select a compute resource to run the VM and click **Next**.

6. On the **Select storage** page, perform the following steps:
   a. Select a VM Storage Policy.
   b. Select the destination datastore for the VM configuration files and virtual disks.
   c. Click **Next**.

7. On the **Select compatibility** page, for the **Compatible with** configuration setting, select **ESXi 6.0 and later** and click **Next**.

8. On the **Select a guest OS** page, perform the following steps:
   a. For **Guest OS Family**, select **Windows**.
   b. For **Guest OS Version**, select **Microsoft Windows Server 2016**.
   c. Click **Next**.

9. On the **Customize hardware** page, configure the VM hardware and click **Next**. When configuring the VM hardware, select the following settings for **New Hard disk** and **New CD/DVD Drive**:
   a. For **New Hard disk**, specify 30 GB or greater.
   b. For **New CD/DVD Drive**, perform the following steps:
      i. Select **Datastore ISO File**.
      ii. Select the ISO file you uploaded to your datastore and click **OK**.
      iii. Enable the **Connect At Power On** checkbox.

10. Review the configuration settings on the **Ready to complete** page and click **Finish**.

**Install Windows Server**

Perform the following steps to install Windows Server on the base VM:

1. After creating the VM, click **Power On** in the **Actions** tab for your VM.

2. Select **Windows Server Standard**.

3. Select **Custom installation**.

4. Complete the installation process, and enter a password for the Administrator user. BOSH later randomizes this password.
Verify OS

⚠️ warning: You must complete the following procedure to verify your OS version before continuing.

Ensure you are using the correct OS version by running the following PowerShell command on the Windows VM:

```
```

The output includes `Version: 10.0.17763`.

Install VMware Tools

Perform the following steps to install VMware Tools on the base VM:

1. Under the VM Summary tab, select Install VMware Tools.
2. Navigate to the `D:` drive and run `setup64.exe`.
   
   **Note:** The VMware Tools install window might appear behind the Command Prompt window.
3. Restart the VM as required to finish the install.

Step 2: Install Windows Updates

This section describes how to install Windows updates on your base Windows VM.

Install Windows Updates

Install Windows updates on the Windows VM using your preferred procedure.

One way to install Windows updates on the Windows VM is by using the `SConfig` utility. Perform the following steps:

1. On the Windows VM, run the `SConfig` utility.
2. Select option number 6, Download and Install Updates.
3. Select A for (A)ll updates.
4. For the Select an option, select (A)ll updates.

You might need to restart the Windows VM while installing updates.

Enable Meltdown Mitigation

⚠️ warning: You must enable Meltdown mitigation. Not enabling Meltdown mitigation can lead to timeout issues while deploying the PASW tile.

Windows Server 2019 should receive the update containing the Meltdown mitigation automatically when you install Windows updates.

After installing Windows update, ensure that the following registry keys are set to enable Meltdown mitigation:
Step 3: Clone the VM

Clone the VM that has the Windows updates installed. Save the original VM so that you can run updates on it in the future.

Perform the following steps:

1. In the vSphere client, right-click the current Windows VM.

2. Select Clone > Clone to Virtual Machine....

3. Ensure that you can create the VM that can be used to create a stemcell for the next Patch Tuesday Monthly Updates.

Step 4: Install Required Software

You might need to specify an explicit execution policy for all of the PowerShell commands in the Step 4: Install Required Software section. You specify an execution policy with the \-ExecutionPolicy flag.

For example:

```
powershell \-ExecutionPolicy Bypass \-Command "Install-CFFeatures"
```

Transfer Files to a Windows VM

Some of the procedures described in the sections below require transferring files to a Windows VM. Many different methods exist to transfer files to a Windows VM, such as folder sharing or the PowerShell Invoke-WebRequest cmdlet. Use whatever method that you prefer.
As an example, the following PowerShell `Invoke-WebRequest` command uses TLS v1.2 to transfer `filename.zip` from `EXAMPLE-URL` to the current location on the Windows VM:

```powershell
Invoke-WebRequest -Uri "EXAMPLE-URL/filename.zip" -OutFile "filename.zip"
```

### Install the BOSH PS Modules

Perform the following steps to install the BOSH PS Modules:

1. Locate the [BOSH PS Modules](#) download for the 2019 stemcell version you want to build, such as 2019.2.

2. Transfer the `bosh-psmodules.zip` file to your Windows VM.

3. Start PowerShell in the Windows VM and run the following command:

   ```powershell
   Unblock-File PATH-TO-BOSH-PSMODULES.ZIP
   
   Where PATH-TO-BOSH-PSMODULES.ZIP is the full path to the location of bosh-psmodules.zip on your Windows VM.
   
   4. Unzip the archive with the following command:

   ```powershell
   Expand-Archive PATH-TO-BOSH-PSMODULES.ZIP C:\Program Files\WindowsPowerShell\Modules
   ```

### Install the Cloud Foundry Diego Cell Requirements

Perform the following steps to install the Cloud Foundry Diego cell requirements:

1. Start PowerShell in the Windows VM and run the following command:

   ```powershell
   Install-CFFeatures
   
   The machine restarts automatically.
   
   2. Apply the recommended ingress and service configuration with the following command:

   ```powershell
   Protect-CFCell
   ```

### Install the BOSH Agent

Perform the following steps to install the BOSH Agent:

1. Locate the [BOSH Agent](#) download for the 2019 stemcell version you want to build, such as 2019.2.

2. Transfer the `agent.zip` file to your Windows VM.

3. Start PowerShell in the Windows VM and run the following command:

   ```powershell
   Unblock-File PATH-TO-AGENT.ZIP
   
   Where PATH-TO-AGENT.ZIP is the full path to the location of the agent.zip file on your Windows VM.
   
   4. Install the BOSH Agent with the following command:

   ```powershell
   Install-Agent -IaaS vsphere -agentZipPath PATH-TO-AGENT.ZIP
   ```

### Install OpenSSH
You can use the `bosh ssh` command on BOSH-deployed Windows VMs if you install the OpenSSH dependency on the Windows VM and then enable it during deploy time. This lets an operator enter into a CMD or PowerShell session on the VM as a user with admin privileges.

Perform the following steps to install OpenSSH:

1. Transfer the [OpenSSH-Win64.zip](#) file to the Windows VM and place it in `C:\provision`.

2. Start PowerShell in the Windows VM and run the following command:

   ```powershell
   Unblock-File 'C:\provision\OpenSSH-Win64.zip'
   ```

3. Install OpenSSH with the following command:

   ```powershell
   Install-SSHD -SSHZipFile 'C:\provision\OpenSSH-Win64.zip'
   ```

4. When configuring the PAS for Windows tile, you must select the BETA: Enable BOSH-native SSH support on all VMs checkbox. For more information, see [Installing and Configuring PAS for Windows](#).

**Optimize and Compress the Disk**

> **Note:** Windows Server stemcells can be large, and can exceed the 10GB upload limit imposed by default by the BOSH Director.

Perform the following steps to reduce the stemcell size:

1. Restart the VM.

2. Start PowerShell in the Windows VM and run the following command to use `dism` to clear unnecessary files:

   ```powershell
   Optimize-Disk
   ```

3. Run the following command to defragment and zero out the disk:

   ```powershell
   Compress-Disk
   ```

**Step 5: Sysprep the System**

This step “syspreps” the system, which ensures that each BOSH VM has a unique identity and applies the appropriate startup configuration at boot time.

The included policies help ensure the uptime and secure operations of the stemcell’s VMs, especially when deployed on PCF.

> **Note:** This step disables services that could cause restarts, such as Windows Automatic Updates. OS restarts are not supported on BOSH-deployed Windows VMs, and the BOSH Director resurrects the VM by destroying and repaving it.

Perform the following steps:

1. Transfer the [LGPO.ZIP](#) file to the Windows VM.

2. Start PowerShell in the Windows VM and run the following command:

   ```powershell
   Expand-Archive PATH-TO-LGPO.ZIP C:\Windows
   ```

3. Run the following command to sysrep the system:

   ```powershell
   Invoke-Sysprep -IaaS vsphere
   [-NewPassword PASSWORD]
   [-Owner OWNER] [-Organization ORGANIZATION]
   ```

> **Note:** All of the flags of `Invoke-Sysprep` except for `-IaaS` are optional.
Where:

- **PASSWORD** is an optional flag that enables you to set a password of your choice. Do not use any special character in the password other than !. For example, `Example12!` is permitted but `ExampleFl2` is not. This is a known issue.
- **OWNER** and **ORGANIZATION** are optional flags. Set them if your organization requires it.

The sysrep command powers off the VM.

⚠️ **warning:** Do not turn the VM back on before completing the procedure in **Step 6: Export the VMDK File**.

### Step 6: Export the VMDK File

Perform the following steps to export the .VMDK file associated with the VM you powered off:

1. In vCenter, right-click the VM and select **Template > Export to OVF Template**.
2. Download the OVA to your local machine. You do not need to include files in the floppy or CD Drive.

   **Note:** You can also download the standalone vsphere client and select **File > Export > Export OVF Template**.

3. Rename the downloaded OVA file to have a `.tar` extension.
4. Expand the TAR archive and locate the VMDK file.

### Step 7: Convert the VMDK File to a BOSH Stemcell

**Note:** This final step typically takes about ten to twenty minutes to complete.

Perform the following steps to convert the VMDK file to a BOSH stemcell:

1. Download the latest release of the **stembuild** utility to your local machine and place the executable in your command-line path.
2. Download **ovftool** to your local machine and place the executable in your command-line path.

   **Note:** On the Windows desktop, **ovftool** is installed by default in `C:\Program Files\VMware\VMware OVF Tool`.

   **stembuild** invokes **ovftool** to convert the disk image to the appropriate stemcell format and apply the proper configuration.

3. Build the stemcell with the following command:

   ```bash
   stembuild package -vmk PATH-TO-VMDK -stemcell-version STEMCELL-VERSION -os 2019
   ```

   Where:
   - **PATH-TO-VMDK** is the path to the VMDK file.
   - **STEMCELL-VERSION** is the 2019 stemcell version you want to build. For example, if you downloaded the BOSH PS Modules and BOSH Agent for the 2019.2 release, then specify `2019.2`.

   **stembuild** creates the stemcell in the directory where you execute it. The file has a `.tgz` extension and a name similar to `bosh-stemcell-2019.2-vsphere-esxi-windows2019-go_agent.tgz`.

   The stemcell is ready for use in conjunction with your BOSH deployment.

### Step 8: Apply Monthly Patch Tuesday Updates

On Patch Tuesday, run Windows Updates on the base image, and then repeat **Step 3: Clone the VM through Step 7: Convert the VMDK File to a BOSH Stemcell**.
Troubleshooting

Garden Windows Logs Suggest Windows Features Not Installed

Symptom

You see the following error in your garden-windows job while deploying Windows Server 2019:

```
Missing required Windows Features:
Web-Webserver, Web-WebSockets, AS-Web-Support,
AS-.NET-Framework, Web-WHC, Web-ASP
Please use the most recent stemcell.
```

Explanation

Install-CFFeatures might not have run successfully.

Solution

Run the following commands in PowerShell on your Windows VM to verify whether Install-CFFeatures ran successfully:

```
Get-WindowsFeature "Containers" | Where InstallState -Eq "Installed"
Get-WindowsFeature "Windows-Defender-Features" | Where InstallState -Eq "Removed"
```
Installing and Configuring PASW

This topic describes how to install and configure the Pivotal Application Service for Windows (PASW) tile. The PASW tile installs Windows cells in your Pivotal Cloud Foundry (PCF) deployment.

Step 1: Confirm Shared PAS Tile Settings

Windows cells installed by the Pivotal Application Service for Windows tile are affected by two settings in the Pivotal Application Service tile. Configure these Pivotal Application Service tile settings as desired:

- **System Logging**: In the System Logging section, if you configure an external syslog aggregator, logs are drained from Windows cells as well as non-Windows cells. See Logging and Metrics for additional information.
- **Disable SSL certificate verification for this environment**: In the Networking section, if you select the Disable SSL certificate verification for this environment checkbox, SSL certificate verification is disabled for Windows cells. See Procedure: Terminate SSL/TLS at HAProxy for additional information.

Note: Only disable SSL certificate verification for development and testing environments. Do not select Disable SSL certificate verification for this environment for production environments.

Step 2: Install the Tile

1. Download the Pivotal Application Service for Windows product file from the product page of Pivotal Network.

2. From the same Pivotal Network page, download the Windows FS Injector tool for your workstation OS.

   The Injector tool, `winfs-injector`, is an executable binary that adds the Windows Server container base image into the product file. This step requires internet access and can take up to 20 minutes.

   Note: You need the `git` and `tar` (BSD) executables on your `%PATH%` to run the `winfs-injector` bin. For example, to use `winfs-injector.exe`, `tar.exe` must be copied to a directory in your `%PATH%`.

3. To add the Windows Server container base image to the product file, run the following command:

   ```
   winfs-injector --input-tile PASW-DOWNLOAD-PATH --output-tile PASW-IMPORTABLE-PATH
   ```

   Where:
   - `PASW-DOWNLOAD-PATH` is the path and filename to the downloaded PASW product file.
   - `PASW-IMPORTABLE-PATH` is the desired output path for the importable product file.

   For example:

   ```
   C:\Users\admin>winfs-injector --input-tile ^
   c:\temp\pas-windows-2.6.0-build.1.pivotal ^
   --output-tile c:\temp\pas-windows-2.6.0-build.1-INJECTED.pivotal
   ```

   For troubleshooting the `winfs-injector`, see Missing Local Certificates for Windows File System Injector.

4. Navigate to the Ops Manager Installation Dashboard and click Import a Product.

5. To add PASW to the Import a Product product list, select the importable `PASW-IMPORTABLE-PATH` file on your workstation.

6. To add PASW to your staging area, click + under the PASW product listing.

Step 3: Configure the Tile

1. Click the newly added PASW tile.
2. Click **Assign AZs and Networks** or **Assign Networks**. The name of the section varies depending on your IaaS.

3. Assign your AZs and networks and click **Save**.

4. Click **VM Options**.

5. Specify your selection for **Manage Administrator Password**.
   - **Use the Windows default password** randomizes the admin password. With this selection, the admin password is not retrievable by an operator. This is the default selection.
   - **Set the password** sets the same admin password for every Windows cell. As a result, this password can be used to access any Windows cell, including, for example, Remote Desktop Protocol (RDP) sessions.

6. (Optional) Select the **BETA: Enable BOSH-native SSH support on all VMs** checkbox to start the Microsoft beta port of the OpenSSH daemon on port 22 on all VMs. Users can SSH onto Windows VMs with the `bosh ssh` command, and enter a CMD terminal as an admin user. They can then run `powershell.exe` to start a PowerShell session.

   **Note:** This feature is beta and not considered production-ready.

7. (Optional) If you want all VMs to support connection through Remote Desktop Protocol (RDP), click **Enable Remote Desktop Protocol**.

8. (Optional) If you want to configure a Key Management Service (KMS) that your volume-licensed Windows cell can register with, perform the following steps:
   a. Click **Enable**
   b. For the Host field, enter the KMS hostname.
   c. For the Port field, enter the port number. The default port number is 1688.
9. Click Save.

10. (Optional) To deploy your PASW application workloads to an isolation segment, click Application Containers and perform the steps in the Assign a Tile to an Isolation Segment section below.

11. (Optional) To configure Windows cells to send Windows Event logs to an external syslog server, click System Logging and perform the steps in the Send Cell Logs to a Syslog Server section.

12. Click Smoke Tests.

   Specify a Cloud Foundry organization and space where smoke tests can run if in the future you delete your Application Service deployment domains.

   Choose where to deploy applications when running the smoke tests *
   - Temporary space within the system organization (This is deleted after smoke tests finish.)
   - Specified org and space (The org and space must have a domain available for routing.)

   Save

13. Choose an org and space where your smoke tests can run.

   - Choosing Temporary space within the system organization deletes the org after smoke tests finish.
   - Choosing Specified org and space requires a domain available for routing and that domain must be accessible to the organization.
     - In the Organization field, enter the Application Service org to use when running tests.
     - In the Space field, enter the Application Service space to use when running tests.
     - In the Domain field, enter the domain that the org has access to for running smoke tests.

14. (Optional) To configure memory and disk overcommit for your Windows Diego cells, click Advanced Features and perform the following steps:

   a. Enter the total desired amount of Diego cell memory in the Cell Memory Capacity (MB) field. For the current cell memory capacity settings, see the Windows Diego cell row on the Resource Config pane.
   b. Enter the total desired amount of Diego cell disk capacity in the Cell Disk Capacity (MB) field. For the current cell disk capacity settings, see the Windows Diego cell row on the Resource Config pane.

   **Note:** Due to the risk of app failure and the deployment-specific nature of disk and memory use, Pivotal has no recommendation about how much, if any, memory or disk space to overcommit.
c. Click **Save**.

15. Click **Errands**.

a. Pivotal recommends that you set the **Install HWC Buildpack Errand** to **On**. This ensures that you receive the most up-to-date HWC Buildpack.

b. Enabling the **Smoke Test Errand** ensures that a smoke test is run against your Application Service installation.

16. Click **Save**.

### Step 4: Configure Tile Resources

To configure your tile resources, perform the following steps:

1. Navigate to the **Resource Config** pane of the PASW tile.

2. Use the dropdown menus to configure **Windows Diego Cell**. For more information, see [Disk Size of Windows Diego Cells](#).

3. Click **Save**.

   **Note:** Provision your **Master Compilation Job** with at least 100 GB of disk space.

### Disk Size of Windows Diego Cells

Windows stemcells in the v2019.x line support ephemeral disks.

The recommended disk size for your Windows Diego cells is as follows:

<table>
<thead>
<tr>
<th>IaaS</th>
<th>Recommended Disk size of Windows Diego cell</th>
</tr>
</thead>
</table>

© Copyright Pivotal Software Inc, 2013-2019 1006 2.5
### Step 5: Upload the Stemcell

1. Go to [Stemcell Library](#).
2. Retrieve the stemcell that you downloaded or created in [Downloading or Creating a Windows Stemcell](#).
3. Follow the steps in [Importing and Managing Stemcells](#) to upload the Windows stemcell to [Pivotal Application Service for Windows](#).

### Step 6: Deploy the Tile

1. Return to the Ops Manager Installation Dashboard.
2. Click [Review Pending Changes](#).
3. Select the PASW tile and review the changes. For more information, see [Reviewing Pending Product Changes](#).
4. Click [Apply Changes](#) to install the PASW tile.

### Step 7: (Optional) Create More Tiles

To run Windows cells in multiple isolation segments, you must create and configure additional PASW tiles. For more information, see [Windows Cells in Isolation Segments](#).

<table>
<thead>
<tr>
<th>Platform</th>
<th>Disk Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>100 GB</td>
</tr>
<tr>
<td>Azure</td>
<td>150 GB</td>
</tr>
<tr>
<td>GCP</td>
<td>150 GB</td>
</tr>
<tr>
<td>vSphere</td>
<td>100 GB</td>
</tr>
</tbody>
</table>

**Note:** If you use vSphere, you must create your own stemcell. The default root disk size of Windows stemcells v2019.x line is 30 GB. Pivotal recommends setting the root disk size of your Windows stemcell for vSphere to 30 GB. For more information, see [Creating a vSphere Windows Stemcell](#).
Windows Cells in Isolation Segments

This topic explains how to run Windows cells in isolation segments, which are compartmentalized resource pools for Diego cells.

Cells in one isolation segment share routing, computing, and logging resources with other cells in the same segment, and do not use resources from other isolation segments.

Prerequisites

To run a Windows cell in a particular isolation segment, the isolation segment must first already exist in the Cloud Controller Database (CCDB) before the Windows cell can be assigned to it.

To create an isolation segment in the CCDB, see: Create an Isolation Segment section of the Managing Isolation Segments topic.

Overview

To run Windows cells in multiple isolation segments, you need to create and install multiple PAS for Windows tiles and configure each to run in a different isolation segment.

See Replicate a Tile for how to create multiple copies of the PAS for Windows tile.

See Assign a Tile to an Isolation Segment for how to associate a PAS for Windows tile with an isolation segment, so that its cells run in that segment.

Replicate a Tile

To make multiple copies of the PAS for Windows tile that you can assign to different isolation segments, use the Replicator tool as follows:

1. Download the Replicator tool from the Pivotal Cloud Foundry Runtime for Windows section of Pivotal Network.
2. Navigate to the directory where you downloaded the Replicator tool.
3. To replicate the tile, run the following command:

   replicator-windows -name "TILE-NAME" -path PATH-TO-ORIGINAL -output PATH-TO-COPY

   Where:
   
   - **TILE-NAME** is a unique name for the new PAS for Windows tile. The name must be ten or fewer characters in length and contain only alphanumeric characters, dashes, underscores, and spaces.
   - **PATH-TO-ORIGINAL** is the absolute path to the original .pivotal PAS for Windows tile you downloaded from Pivotal Network.
   - **PATH-TO-COPY** is the absolute path for the copy of the .pivotal PAS for Windows tile that the Replicator tool produces.

   For example:

   ```
   C:\Users\admin> replicator-windows -name "Shiny Tile" ^
   -path c:\downloads\TileOriginal.pivotal ^
   -output c:\temp\TileCopy.pivotal
   ```

4. Install and configure the Windows isolation segment, using the new .pivotal file and following the procedures in this topic, starting with the Import a Product step of Step 2: Install the Tile.

Assign a Tile to an Isolation Segment

To assign a PAS for Windows tile to an isolation segment, perform the following steps:

1. Open the Application Containers pane.
2. Under **Segment Name**, enter the name for the isolation segment to associate the tile with. If you are creating a new isolation segment, ensure that this name is unique across your PCF deployment.

3. Click **Save**.
Migrating Apps to PAS for Windows

This topic describes the process of migrating apps running on Pivotal Application Service (PAS) for Windows 2012R2 cells to run on PAS for Windows cells, which run Windows Server 2019.

**Note:** PAS for Windows 2012R2 has reached its End of Availability (EoA) and is no longer generally available.

Pivotal recommends you use the blue-green deployment method for high availability. For more information about blue-green deployments, see [Using Blue-Green Deployment to Reduce Downtime and Risk](#).

### Step 1: Install and Deploy PAS for Windows Tile

To install and deploy the PAS for Windows tile, follow steps 1 and 2 of [Installing and Configuring PAS for Windows](#).

### Step 2: Push App to PAS for Windows Cells

Perform the following steps to redeploys a running app with zero downtime using the blue-green method:

1. To log in to the Cloud Foundry Command Line Interface (cf CLI), run the following command:
   ```bash
cf login
   ```

2. Choose your org and space.

3. Navigate to the location of your app.

4. To find the name of the existing PAS for Windows 2012R2 app you are migrating to PAS for Windows, run the following command:
   ```bash
cf apps
   ```

5. Create a name for the replacement PAS for Windows app. Pivotal recommends you append `-green` to your existing app name, using the following format:
   ```bash
   APP-NAME-green
   ```
   Where `APP-NAME` is the existing PAS for Windows 2012R2 app's name.

6. To push your app to a PAS for Windows Diego cell, using your newly created name, run the following command:
   ```bash
cf push APP-NAME-GREEN -s windows -b BUILDPACK -a HOSTNAME --no-start --no-route
   ```
   Where:
   - **APP-NAME-GREEN** is the newly created “green” name for your app.
   - **BUILDPACK** is your custom buildpack. Specify the buildpack either by name or GitHub URL with an optional branch or tag.
   - **HOSTNAME** is the name of your app's subdomain. For example, if `example.com` is your domain and you want the URL to your app to be `http://my-app.example.com`, then specify `my-app` as the `HOSTNAME`.

For example:
```bash
C:\Users\admin> cf push ExampleApp-green -s windows ^  -b https://github.com/cloudfoundry/binary-buildpack.git ^  -a my-app --no-start --no-route
```

**Note:** The `--no-start` and `--no-route` parameters included in this `cf push` command are required for this procedure. `--no-start` is used to create the instance VMs and not start the app, and `--no-route` is used to prevent the push command from automatically mapping a route to the app. For additional information on `cf push`, see [cf push](#).

---

© Copyright Pivotal Software Inc, 2013-2019
7. To configure the router so all incoming requests go to both [APP-NAME] and [APP-NAME-green], run the following command:

```
cf map-route APP-NAME-green DOMAIN -n HOSTNAME
```

Where:
- [APP-NAME-green] is the name of the new “green” version of your app.
- DOMAIN is your domain name, for example, example.com.
- HOSTNAME is the name of your app’s subdomain. For example, if example.com is your domain and you want the URL to your app to be http://my-app.example.com, then specify my-app as the HOSTNAME.

> **Note:** When using the `cf map-route` command, you must specify domain name after specifying app name.

For additional information on `cf map-route`, see `cf map-route`.

8. To start the green app, run the following command:

```
cf start APP-NAME-green
```

Where [APP-NAME-green] is the name of the new “green” version of your app.

9. To confirm that both your [APP-NAME] and [APP-NAME-green] apps are running, run the following command:

```
cf apps
```

If you experience a problem, see Troubleshooting Application Deployment and Health.

10. To unmap the original app’s route, run the following command:

```
cf unmap-route APP-NAME DOMAIN -n HOSTNAME
```

Where:
- [APP-NAME] is the name of the existing version of your app you want to replace with [APP-NAME-green].
- DOMAIN is your domain name, for example, example.com.
- HOSTNAME is the name of your app’s subdomain. For example, if example.com is your domain and the existing app is currently accessed using the URL http://my-app.example.com, then specify my-app as the HOSTNAME.

For additional information on `cf-unmap-route`, see `cf unmap-route`.

### Step 3: Delete App from Windows 2012R2 Server Cells

1. To delete the original app, run the following command:

```
cf delete APP-NAME
```

Where [APP-NAME] is the name of the app that you have replaced with [APP-NAME-green].

> **Note:** To also delete any mapped routes, run the command with the `-r` flag.

For additional information on `cf delete`, see `cf delete`.

### Step 4: (Optional) Uninstall Old Tile

Once you have migrated all of your apps and you are no longer using the Windows 2012R2 tile, a PCF operator can perform the following steps:

1. From the Installation Dashboard, click the trash icon on the tile to remove that product. In the Delete Product dialog box that appears, click Confirm.

2. Click Review Pending Changes.

3. Select the PAS for Windows tile and review the changes. For more information, see Reviewing Pending Product Changes.
4. Click **Apply Changes**.

**Note:** After you delete a product, the product tile is removed from the installation and the Installation Dashboard. However, the product appears in the Available Products view.
Upgrading PAS for Windows and Windows Cells

This topic describes how to upgrade the Pivotal Application Service (PAS) for Windows tile and update the Windows stemcell.

For how developers can migrate apps to PAS for Windows from either PAS for Windows 2012R2 or PCF Runtime for Windows, see Migrating Apps to PAS for Windows.

Rotating Credentials

The PAS tile handles all of the credentials for PAS for Windows.

To rotate your credentials in PAS for Windows, re-deploy PAS and PAS for Windows following the steps below:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click Review Pending Changes.
3. Select the PAS and PAS for Windows tiles and review the changes. For more information, see Reviewing Pending Product Changes.
4. Click Apply Changes.

Upgrade PAS for Windows

To upgrade PAS for Windows:

1. Follow the instructions Upgrading PCF Products with the latest PAS for Windows product download from Pivotal Network.
2. If necessary, configure the product. For more information about configuring PAS for Windows, see the Installing and Configuring PAS for Windows topic.
3. Click Review Pending Changes.
4. Select the PAS for Windows tile and review the changes. For more information, see Reviewing Pending Product Changes.
5. Click Apply Changes.

Upgrade the Windows Stemcell

1. Retrieve the stemcell by following the steps below for your IaaS:
   - For vSphere, you must build your own stemcell.
   - For AWS, GCP, and Azure, navigate to the Stemcells for PCF (Windows Server) section of Pivotal Network.
2. Go to Stemcell Library.
3. Click Import Stemcell to import the stemcell file.
4. When prompted, enable the Ops Manager product checkbox to stage your stemcell.
5. Click Apply Stemcell to Products.

For more information about Windows stemcells, see Downloading or Creating Windows Stemcells.
Using SMB Volumes in .NET Apps

This topic describes how to use Server Message Block (SMB) Volumes in a .NET Application. In this example, we create an application that reads and writes to a `note.txt` file within an SMB Volume share.

Prerequisites

The following is required to use SMB Volumes in your .NET App:

- Visual Studio
- An Azure account
- Pivotal Application Service for Windows (PASW)
- An accessible SMB Volume in Azure Cloud with a UNC path
  - If you do not have an accessible SMB Volume, refer to Create a file share in Azure Files in the Azure documentation.
- An existing .NET Framework application
- SMB Username
- SMB Password
- Windows 1803.x or Windows 1709.x Stemcell

Using SMB Volumes in .NET Apps with Steeltoe

Steeltoe is a set of libraries that help your team write cloud native applications. Steeltoe contains functionality for mounting SMB Volumes. Using Steeltoe requires modifying your application code. For more information, see the Steeltoe Documentation.

If you are using Steeltoe, add the following environment variables to your Config Server Provider:

- `SMB_PATH`
- `SMB_USERNAME`
- `SMB_PASSWORD`

Where:

- `SMB_PATH` is the UNC path to your SMB.
  - If you are using the Windows 1709 Stemcell, your UNC is the IP of the machine you are using.
  - If you are using the Windows 1803 Stemcell, your UNC is the FQDN of the machine you are using.
- `SMB_USERNAME` is your Azure account username.
- `SMB_PASSWORD` is your Azure account password.

For more information on adding environment variables to your Config Server Provider, see Config Server Provider in the Steeltoe documentation.

After adding your environment variables to your Config Server Provider, proceed to SMB Mounting.

Using SMB Volumes in .NET Apps with Your Batch Profile

If you do not want to modify your application code to include Steeltoe, you can utilize SMB Mounting without Steeltoe.

1. Retrieve the UNC of your existing SMB share.
   a. If you are using the Windows 1709.x Stemcell, your UNC is your machine’s IP address.
      i. To look up your machine’s IP address, you can run the `nslookup` command. For more information, see nslookup in Microsoft’s Commands by Server Role Documentation.
   b. If you are using the Windows 1803 Stemcell, your UNC is your machine’s FQDN.
2. Create a `.profile.bat` file in your .NET app’s root directory.

3. Use Apps Manager or the Cloud Foundry Command Line Interface (cf CLI) to set the following environment variables:
   
   ```
   SMB_PATH
   SMB_USERNAME
   SMB_PASSWORD
   ```

4. Add the following command to your `.profile.bat` file:
   ```
   net use z: %SMB-PATH% %SMB-PASSWORD% /USER:SMB-USERNAME%
   ```

5. Proceed to SMB Mounting.

SMB Mounting

1. Using CF CLI or Apps Manager, add `SMB_PATH`, `SMB_USERNAME`, and `SMB_PASSWORD` as environment variables to your local computer or the computer you are deploying using for deploying your application.
   - If you are using SMB Volumes with your batch profile and have already added these environment variables, ignore this step.

   ![Note: If Visual Studio does not detect these new environment variables, restart Visual Studio.]

2. In Visual Studio, create a new file in your solution named `SMBConfiguration.cs`. This file is the single representation of your SMB Volume configuration and reads the connection data from the environment variables you established previously.

   ```csharp
   // SMBConfiguration.cs
   using System;

   namespace NetFrameworkApp.Controllers
   {
   public class SMBConfiguration
   {
   public String GetSharePath()
   {
   return Environment.GetEnvironmentVariable("SMB_PATH");
   }

   public String GetUserName()
   {
   return Environment.GetEnvironmentVariable("SMB_USERNAME");
   }

   public String GetPassword()
   {
   return Environment.GetEnvironmentVariable("SMB_PASSWORD");
   }
   }
   }
   ```

3. In Visual Studio, create a new MVC Controller named `NoteController`. This file creates a controller endpoint that reads a the example note file. For more information on creating a controller, see [Add a controller to an ASP.NET Core MVC app](https://aka.ms/aspnetcore/ct) in the Microsoft Documentation.

4. In the command line, add `Steeltoe.Common` and `Steeltoe.Common.Net` to your application with a Package Manager of your choice. If you are not using Steeltoe, ignore this step.

5. Edit `NoteController.cs` to read from a file named `note.txt`, which does not exist yet but the `FileMode.OpenOrCreate` method creates that file. Refer to the example code snippet below, which reads the contents of the note file and stashes the `note.txt` content in the Viewbag. If you are not using Steeltoe, ignore the reference to `Steeltoe.Common.Net` in the following code snippet.

   ```csharp
   // NoteController.cs
   using System;
   using System.Collections.Generic;
   using System.IO;
   using System.Linq;
   using System.Net;
   using System.Web;
   using System.Web.Mvc;
   using Steeltoe.Common.Net;
   ```

© Copyright Pivotal Software Inc, 2013-2019 1015 2.5
namespace NetFrameworkApp.Controllers
{
    public class NoteController : Controller
    {
        SMBConfiguration configuration = new SMBConfiguration();
        public ActionResult Index()
        {
            var credential = new NetworkCredential(configuration.GetUserName(), configuration.GetPassword());
            using (var share = new WindowsNetworkFileShare(configuration.GetSharePath(), credential))
            using (var inputStream = new FileStream(Path.Combine(configuration.GetSharePath(), "note.txt"), FileMode.OpenOrCreate))
            using (var streamReader = new StreamReader(inputStream))
            {
                // Never display raw user input as HTML. Do not do this in production code.
                ViewBag.Note = streamReader.ReadToEnd();
            }
            return View();
        }
    }
}

6. In Visual Studio, create a subdirectory in Views named \Note.\

7. In Visual Studio, create a new View named \Index. For more information on Views, see [Views in ASP.NET Core MVC](https://docs.microsoft.com/en-us/aspnet/core/mvc/views) and [Add a view to an ASP.NET Core MVC app](https://docs.microsoft.com/en-us/aspnet/core/mvc/views) in the Microsoft Documentation.\

8. In Visual Studio, create a \Index.cshtml file that contains the following:

   // Index.cshtml

   @ViewBag.Note

   Running the application now would show an empty page with no errors.

9. Modify the \Index.cshtml file to contain a form. This form posts to a yet to be created update endpoint and also displays our note inside a text area.

   // Index.cshtml

   ...
   <form action="https://docs.pivotal.io/note/update" method="post">
   <textarea name="note">@ViewBag.Note</textarea>
   <div>
   <button type="submit">Update</button>
   </div>
   </form>

10. Modify the \NoteController.cs to have an update endpoint. This endpoint saves the data posted to the endpoint back into the \note.txt. Refer to the example code snippet in the following step.\

11. Modify the \NoteController.cs to include the ControllerBase class \RedirectToAction method, which redirects to the index page so the user can see what was just saved. For more information on the ControllerBase class, see the [Microsoft .NET API Browser Documentation](https://docs.microsoft.com/en-us/dotnet/api/system.mvc.controllerbase).\

   Refer to the example code snippet below.

   // NoteController.cs

   namespace NetFrameworkApp.Controllers
   {
   public class NoteController : Controller
   {
   ...
   [HttpPost]
   public ActionResult Update(String note)
   {
   var credential = new NetworkCredential(configuration.GetUserName(), configuration.GetPassword());
   using (var share = new WindowsNetworkFileShare(configuration.GetSharePath(), credential))
   {
Known Issue

Inability to SSH Into App Instance

Despite an SMB mapping being successfully created, trying to `cf ssh` into that app instance to access the created mapping results in an unspecified path error.
Troubleshooting Windows Cells

This topic describes how to troubleshoot Windows Diego cells deployed by Pivotal Application Service (PAS) for Windows.

Installation Issues

This section describes issues that may occur during the installation process.

Missing Local Certificates for Windows File System Injector

Symptom

You run the `winfs-injector` and see the following error about certificates:

```
```

Explanation

Local certificates are needed to communicate with Docker Hub.

Solution

Install the necessary certificates on your local machine. On Ubuntu, you can install certificates with the `ca-certificates` package.

Outdated Version for Windows File System Injector

Symptom

You run the `winfs-injector` and see the following error about a missing file or directory:

```
open ...windows2016fs-release/VERSION: no such file or directory
```

Explanation

You are using an outdated version of the `winfs-injector`.

Solution

From Pivotal Network, download the recommended version of PAS for Windows File System Injector tool for the tile.

Missing Container Image

Symptom

You click the + icon in Ops Manager to add the PAS for Windows tile to the Installation Dashboard, and you see the error:

```
is invalid: has an invalid release 'windows2016fs' specified for job type 'Windows Diego Cell'
```

Explanation

The product file you are trying to upload does not contain the Windows Server container base image.
Solution

1. Delete the product file listing from Ops Manager by clicking its trash can icon under Import a Product.

2. Follow the PAS for Windows installation instructions to run the winfs-injector tool locally on the product file. This step requires internet access, can take up to 20 minutes, and adds the Windows Server container base image to the product file.

3. Click Import a Product to upload the “injected” product file.

4. Click the + icon next to the product listing to add the PAS for Windows tile to the Installation Dashboard.

Windows Cell Log Types

Windows cells generate two types of logs:

- **BOSH job logs**, such as rep_windows and consul_agent_windows. These logs stream to the syslog server configured in the PAS tile System Logging pane, along with other PCF component logs. The names of these BOSH job logs correspond to the names of the logs emitted by Linux Diego cells.

- **Windows Event logs**. These stream to the syslog server configured in the PAS for Windows System Logging pane and are downloadable through Ops Manager, as described below.

Access Windows Event Logs

PCF operators can access log messages from Windows Diego cells in two ways:

- Configure PAS for Windows to send all Windows cell logs to an external syslog server.
- Download archived logs from each Windows cell individually.

Send Cell Logs to a Syslog Server

To forward Windows cell log messages to an external syslog server, complete the following steps:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the PAS for Windows tile.

3. Under the **Settings** tab, select the **System Logging** pane.
4. Under **Enable Syslog for VM logs?**, click **Enable**.

5. Under **Address**, enter the IP address of your syslog server.

6. Under **Port**, enter the port of your syslog server. The typical port for a syslog server is **514**.

   **Note:** The host must be reachable from the PAS network. Ensure your syslog server listens on external interfaces.

7. Under **Protocol**, select the transport protocol to use when forwarding logs.

8. Click **Save**.

### Download Cell Logs

Perform the following steps to retrieve the logs from a Windows cell:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the **PAS for Windows** tile.

3. Click the **Status** tab.

4. Under the **Logs** column, click the download icon for the Windows cell you want to retrieve logs from.

5. Click the **Logs** tab.

6. When the logs are ready, click the filename to download them.

7. Unzip the file to examine the contents. Each component on the cell has its own logs directory:

   - `/consul_agent_windows/`
   - `/garden-windows/`
   - `/metron_agent_windows/`
   - `/rep_windows/`

---

**Connect to a Windows Cell**
Perform the following steps to connect to a Windows cell to run diagnostics:

1. Download and install a Remote Desktop Protocol (RDP) client.
   - For Mac OS X, download the Microsoft Remote Desktop app from the [Mac App Store](https://mac.appstore.com).
   - For Windows, download the Microsoft Remote Desktop app from [Microsoft](https://go.microsoft.com).
   - For Linux/UNIX, download a RDP client like [rdesktop](https://rdesktop.net).

2. Follow the steps in the Log into BOSH section of the Advanced Troubleshooting with the BOSH CLI topic to log in to your BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.

3. To retrieve the IP address of your Windows cell using the BOSH CLI, run the following command:

   ```
   bosh -e ENV-NAME -d DEPLOYMENT-NAME
   ```

   Where:
   - `ENV-NAME` is the alias you assigned to your BOSH Director.
   - `DEPLOYMENT-NAME` is your deployment’s name.

   For example:

   ```
   c:\Users\admin> bosh -e my-environ -d garden-windows
   Using environment '192.0.2.6' as client 'admin'
   ```

4. Retrieve the Administrator password for your Windows cell by following the steps for your IaaS:
   - On vSphere, this is the value of `WINDOWS_PASSWORD` in the `consumer-vars.yml` file you used to previously build a stemcell.
   - On Amazon Web Services (AWS), navigate to the AWS EC2 console. Right-click on your Windows cell and select Get Windows Password from the drop-down menu. Provide the local path to the `.ops_mgr.pem` private key file you used when installing Ops Manager and click Decrypt password to obtain the Administrator password for your Windows cell.
   - On Google Cloud Platform (GCP), navigate to the Compute Engine Dashboard. Under VM Instances, select the instance of the Windows VM. At the top of the page, click on Create or reset Windows password. When prompted, enter “Administrator” under Username and click Set. You will receive a one-time password for the Windows cell.
   - You cannot RDP into Windows cells on Azure.

5. Open your RDP client. The examples below use the Microsoft Remote Desktop app.

   ![Remote Desktop Client](image)

   6. Click **New** and enter your connection information:
      - **Connection name**: Enter a name for this connection.
      - **PC name**: Enter the IP address of your Windows cell.
7. To mount a directory on your local machine as a drive in the Windows cell, perform the following steps:

a. From the same Edit Remote Desktops window as above, click Redirection.

b. Click the plus icon at the bottom left.

c. For Name, enter the name of the drive as it will appear in the Windows cell. For Path, enter the path of the local directory.

d. Click OK.

8. Close the Edit Remote Desktops window and double-click the newly added connection under My Desktops to open a RDP connection to the Windows cell.

9. In the RDP session, you can use the Consul CLI to diagnose problems with your Windows cell.

Consul CLI

Perform the following steps to use the Consul CLI on your Windows cell to diagnose problems with your Consul cluster:

1. In your RDP session, open a PowerShell window.

2. To change to the directory containing the Consul CLI binary, run the following:

   ```
   cd CONSUL-CLI-DIR/bin
   ```

   Where `CONSUL-CLI-DIR` is the Consul CLI package's directory path.

   For example:

   ```
   PS C:\Users\admin> cd C:\var\vcap\packages\consul-windows\bin
   ```

3. To list the members of your Consul cluster, run the following:

   ```
   consul.exe members
   ```

   For example:
<table>
<thead>
<tr>
<th>Node</th>
<th>Address</th>
<th>Status</th>
<th>Type</th>
<th>Build</th>
<th>Protocol</th>
<th>DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell-windows-0</td>
<td>10.0.0.111:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>cloud-controller-0</td>
<td>10.0.0.94:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>cloud-controller-worker-0</td>
<td>10.0.0.99:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>consul-server-0</td>
<td>10.0.0.96:8301</td>
<td>alive</td>
<td>server</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>diego-brain-0</td>
<td>10.0.0.109:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>diego-cell-0</td>
<td>10.0.0.103:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>diego-cell-1</td>
<td>10.0.0.104:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>diego-cell-2</td>
<td>10.0.0.107:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>diego-database-0</td>
<td>10.0.0.92:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>ha-proxy-0</td>
<td>10.0.0.254:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>nfs-server-0</td>
<td>10.0.0.100:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>router-0</td>
<td>10.0.0.105:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
<tr>
<td>uaa-0</td>
<td>10.0.0.93:8301</td>
<td>alive</td>
<td>client</td>
<td>0.6.4</td>
<td>2</td>
<td>dc1</td>
</tr>
</tbody>
</table>

4. Examine the output to ensure that the `cell-windows-0` service is registered in the Consul cluster and is `alive`. Otherwise, your Windows cell cannot communicate with your PCF deployment and developers cannot push .NET apps to the Windows cell. Check the configuration of your Consul cluster, and ensure that your certificates are not missing or misconfigured.
Using Apps Manager

The web-based Apps Manager application helps you manage users, organizations, spaces, and applications.

Table of Contents

- Getting Started with Apps Manager
- Managing Orgs and Spaces Using Apps Manager
- Managing User Roles with Apps Manager
- Managing Apps and Service Instances Using Apps Manager
- Viewing ASGs in Apps Manager
- Monitoring Instance Usage with Apps Manager
- Configuring Spring Boot Actuator Endpoints for Apps Manager
- Using Spring Boot Actuator Endpoints with Apps Manager
Getting Started with Apps Manager

This topic describes the functions and scope of Apps Manager, a web-based tool for managing Pivotal Application Service (PAS) organizations, spaces, applications, services, and users.

Scope

Apps Manager provides a visual interface for performing the following subset of functions available through the Cloud Foundry Command Line Interface (cf CLI):

- **Orgs**: You can create and manage orgs.
- **Spaces**: You can create, manage, and delete spaces.
- **Apps**: You can scale apps, bind apps to services, manage environment variables and routes, view logs and usage information, start and stop apps, and delete apps.
- **Services**: You can bind services to apps, unbind services from apps, choose and edit service plans, and rename and delete service instances.
- **Users**: You can invite new users, manage user roles, and delete users.

To access Apps Manager as the Admin user, see the [Logging in to Apps Manager](#) topic.

Browser Support

Apps Manager is compatible with the latest major versions of the following browsers:

- [Apple Safari](#)
- [Google Chrome](#)
- [Microsoft Edge](#)
- [Microsoft Internet Explorer](#)
- [Mozilla Firefox](#)

Pivotal recommends using Chrome, Firefox, Edge, or Safari for the best Apps Manager experience.

About Permissions

Your ability to perform actions in Apps Manager depends on your user role and the feature flags that the Admin sets.

The table below shows the relationship between specific org and space management actions and the non-Admin user roles who can perform them. A non-Admin user must be a member of the org and space to perform these actions.

Admin users can perform all of these actions using either the cf CLI or by logging into Apps Manager as an Org Manager, using the UAA Admin credentials.

Space Managers assign and remove users from spaces by setting andunsetting their roles within the space.

<table>
<thead>
<tr>
<th>Action</th>
<th>CLI command</th>
<th>Org Manager</th>
<th>Space Manager</th>
<th>Org Auditor, Space Developer, or Space Auditor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an org</td>
<td>create-org</td>
<td>†</td>
<td>†</td>
<td>†</td>
</tr>
<tr>
<td>Delete an org</td>
<td>delete-org</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rename an org</td>
<td>rename-org</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>View org members</td>
<td>org-users</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Assign user a role in org</td>
<td>set-org-role</td>
<td>†</td>
<td>†</td>
<td>No</td>
</tr>
<tr>
<td>Remove org role from user</td>
<td>unset-org-role</td>
<td>†</td>
<td>†</td>
<td>No</td>
</tr>
<tr>
<td>View space members</td>
<td>space-users</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Assign user a role in space</td>
<td>set-space-role</td>
<td>†</td>
<td>†</td>
<td>No</td>
</tr>
<tr>
<td>Remove space role from user</td>
<td>unset-space-role</td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

† Defaults to no. Yes if feature flag `user_org_creation` is set to `true`.

‡ Defaults to no. Yes if feature flags `set_roles_by_username` and `unset_roles_by_username` are set to `true`.

© Copyright Pivotal Software Inc, 2013-2019
Managing Orgs and Spaces Using Apps Manager

This topic discusses how to view and manage orgs and spaces in Apps Manager.

Note: To manage a space, you must have Space Manager permissions in that space.

To perform the following steps, you must first log in to Apps Manager with an account that has adequate permissions. See the About Permissions topic for more information.

Manage an Org

The following sections describe how to manage your orgs using the org management interface in Apps Manager. You can navigate to your org from the Home page or through the global search.

For details about each tab in the org page, see the following sections.

Spaces

In the Spaces tab, you can do the following:

- See the following information for each space:
  - Name
  - Number of apps
  - Number of apps in running, stopped, and crashed states
  - Number of services
  - Memory quota usage
- Create a new space

Domains

In the Domains tab, you can do the following:

- See each Domain in your org.
- See the Type associated with each domain. A type is either Globally Shared or Privately Shared. For more information about globally and locally shared domains, see Shared Domains and Private Domains.
- Delete a domain by clicking the vertical three-dot icon and selecting Delete. You must have permission to manage the domain to delete it.
- Create a new private domain by clicking Add a Domain. You must have Org Manager permissions to create a private domain.

Note: To create globally shared domains, you must have Admin permissions and use the Cloud Foundry Command Line Interface (cf CLI). For more information, see Create a Shared Domain.

See the following screenshot for an example Domains interface with Org Manager permissions:
Members

In the Members tab, you can do the following:

- See member email addresses
- Toggle member permissions
- Remove member from org
- Invite a new member

Settings

In the Settings tab, you can do the following:

- Change the org name
- See spaces assigned to isolation segments. For more information, see Isolation Segments.
- Delete the org. You must have admin permissions to delete an org.

Manage a Space

The space page displays the apps, service instances, and routes associated with the selected space.
Apps
The Apps tab shows the Status, the Name, the number of Instances, the amount of Memory available, the time since the Last Push, and the Route for each app.

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Instances</th>
<th>Memory</th>
<th>Last Push</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>hello-spring-cloud</td>
<td>1</td>
<td>256 MB</td>
<td>4 hours ago</td>
<td><a href="https://docs-team-spring-app.apps.oo">https://docs-team-spring-app.apps.oo</a>...</td>
</tr>
<tr>
<td>Running</td>
<td>my-super-app</td>
<td>1</td>
<td>20 MB</td>
<td>4 hours ago</td>
<td>no bound route</td>
</tr>
</tbody>
</table>

Services
The Services list shows the Service, the Name, the number of Bound Apps, and the Plan for each service instance. If you want to add a service to your space, click Add Service. For more information about configuring services, see the Services Overview topic.

<table>
<thead>
<tr>
<th>Service</th>
<th>Name</th>
<th>Bound Apps</th>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Autoscaler</td>
<td>my-autoscaler</td>
<td>0</td>
<td>Free + Standard</td>
</tr>
</tbody>
</table>

Routes
From the Routes tab, you can view and delete routes associated with your space. For each Route, the tab shows Route Service and Bound Apps. Refer to the Route Services topic to learn more about route services.

<table>
<thead>
<tr>
<th>Route</th>
<th>Route Service</th>
<th>Bound Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs-team-broker-app.apps.ogie-boogie...</td>
<td>my-service</td>
<td>pong-matcher</td>
</tr>
<tr>
<td>docs-team-spring-app.apps.ogie-boogie...</td>
<td>no bound service</td>
<td>hello-spring-cloud</td>
</tr>
</tbody>
</table>

Settings
From the Settings tab, you can do the following:

- Modify the space name by entering a new name and clicking Update.
- View the Application Security Groups (ASGs) associated with the space in the Security Groups section.
- View the space’s isolation segment assignment in the Isolation Segment Assignment section. For more information, see Isolation Segments.
- Add metadata to the space. See Add Metadata.
- Delete the space by clicking Delete Space.
Add Metadata

You can add metadata, including labels and annotations, to apps and spaces using Apps Manager. For more information about adding metadata to apps using Apps Manager, see Managing Apps and Service Instances Using Apps Manager.

For more information about adding metadata to objects in PAS, see Using Metadata.

To add labels or annotations to a space using Apps Manager, do the following:

1. Click the Settings tab of the desired space.
2. For Labels, provide a name and value pair. Or, select Enter JSON to enter the label in JSON.
3. For Annotations, provide a name and value pair. Or, select Enter JSON to enter the annotation in JSON.
4. To add more labels or annotations, click + next to the Labels or Annotations field.
5. Click Update Metadata.
Managing User Roles with Apps Manager

Cloud Foundry uses role-based access control, with each role granting the permissions in either an org or an application space.

A user account can be assigned one or more roles.

The combination of these roles defines the actions a user can perform in an org and within specific app spaces in that org.

To view the actions that each role allows, see the Organizations, Spaces, Roles, and Permissions topic. For example, to assign roles to user accounts in a space, you must have Space Manager role assigned to the user in that space.

You can also modify permissions for existing users by adding or removing the roles associated with the user account. User roles are assigned on a per-space basis, so you must modify the user account for each space that you want to change.

Admins, Org Managers, and Space Managers can assign user roles with Apps Manager or with the Cloud Foundry Command Line Interface (cf CLI). For more information, see the Users and Roles section of the Getting Started with the cf CLI topic.

Manage Org Roles

Valid org roles are Organization Manager and Organization Auditor.

To grant or revoke org roles, follow the steps below.

1. Go to the Home page
2. Select an org.
3. Click the Members tab. Edit the roles assigned to each user by selecting or clearing the checkboxes under each user role. Apps Manager saves your changes automatically.
4. The Members panel displays all members of the org. Select a checkbox to grant an org role to a user, or clear a checkbox to revoke a role from a user.

Manage App Space Roles

Valid app space roles are Space Manager, Space Developer, and Space Auditor.

To grant or revoke app space roles, do the following:

1. Go to the page for a space
2. Select the Members tab. The Members panel displays all members of the space.
3. Select a checkbox to grant an app space role to a user, or clear a checkbox to revoke a role from a user.
   * Space Managers can invite and manage users and enable features for a given space. Assign this role to managers or other users who need to administer the account.
   * Space Developers can create, delete, and manage applications and services, and have full access to all usage reports and logs. Space Developers can also edit applications, including the number of instances and memory footprint. Assign this role to app developers or other users who need to interact with applications and services.
   * Space Auditors have view-only access to all space information, settings, reports, and logs. Assign this role to users who need to view but not edit the application space.

Invite New Users
1. On the Org dashboard, click the Members tab.

2. Click Invite New Members. The Invite New Team Member(s) form appears.

3. In the Add Email Addresses text field, enter the email addresses of the users that you want to invite. Enter multiple email addresses as a comma-delimited list.

4. The Assign Org Roles and Assign Space Roles tables list the current org and available spaces with checkboxes corresponding to each possible user role. Select the checkboxes that correspond to the permissions that you want to grant to the invited users.

5. Click Send Invite. The Apps Manager sends an email containing an invitation link to each email address that you specified.

Remove a User From an Org

Removing a user from org also removes them from all spaces in the org.

1. On the Org dashboard, click the Members tab.

2. Locate the user account that you want to remove.

3. Under the user's email address, click on the Remove User link. A warning dialog appears.
4. Click Remove to confirm user account deletion from the org.

Remove a User From a Space

1. Go to the page for a space

2. Select the Members tab. The Members panel displays all members of the space

3. Locate the user account that you want to remove.

4. Under the user’s email address, click on the Remove User link. A warning dialog appears.

5. Click Remove to confirm user account deletion from the space.
Managing Apps and Service Instances Using Apps Manager

This topic discusses how to view and manage apps and service instances associated with a space using Apps Manager.

To perform the following steps, you must first log in to Apps Manager with an account that has adequate permissions. See the About Permissions topic for more information.

Manage an App

On the space page, click the app you want to manage. You can also search for an app by entering its name in the search bar.

From the app page, you can scale apps, bind apps to services, manage environment variables and routes, view logs and usage information, start and stop apps, and delete apps.

Start, Stop, or Restage an App

The start, stop, and restage buttons are located next to the name of your app. See the diagram below:
To perform actions with the three buttons, see the table below:

<table>
<thead>
<tr>
<th>To…</th>
<th>Do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop an app</td>
<td>Click the (stop) button</td>
</tr>
<tr>
<td>Restart a stopped or running app</td>
<td>Click the (restart) button</td>
</tr>
<tr>
<td>Restage an app</td>
<td>Click the (restage) button</td>
</tr>
</tbody>
</table>

For more information about starting, stopping, and restaging apps, but with the Cloud Foundry Command Line Interface (cf CLI), see Starting, Restarting, and Restaging Applications.

Scale an App

From the app Overview pane, Space Developers can scale an app manually or configure App Autoscaler to scale it automatically.

Scale an App Manually

1. Click Scale to open the “Scale App” dialog.
2. Adjust the number of Instances, the Memory Limit, and the Disk Limit as desired.

3. Click Apply Changes.

Configure App Autoscaler
1. Use **Autoscaling** to enable App Autoscaler.

2. Click **Manage Autoscaling** to open App Autoscaler.

3. See the **Configure Autoscaling for an App** section of the *Scaling an Application Using Autoscaler* topic for how to configure your App Autoscaler to scale automatically based on rules or a schedule.

**Bind or Unbind a Service**

**Note:** For services that use asynchronous bindings, Apps Manager displays the status of the service while the bind is still pending. Asynchronous bindings provide more flexibility for services that require additional time before returning a successful bind response.

1. Click **Services**.

2. To bind your app to a service, click **Bind Service**.

3. To bind your app to an existing service instance, do the following:
   a. Click **Bind Service**.

   b. Under **Service to Bind**, select the service instance from the dropdown menu.
   c. (Optional) Enter a binding name under **Binding Name** in all lowercase letters. For example, `bind-usr-serv`.
   d. (Optional) Specify additional parameters under **Add Parameters**.
   e. Click **Bind**.

4. To bind your app to a new service instance, do the following:
   a. Click **New Service**.
b. Click the service.

![Create a new service](image)

```
Create a new service
App Autoscaler
Scales bound applications in response to load (beta)

Standard
free
- Show Details

BACK SELECT PLAN
```

c. Select a plan and click Select Plan.

![Create a new service](image)

```
Create a new service
App Autoscaler
standard - free

Instance Name
my-autoscaler

Add to Space
development

Bind to App (Optional)
cf.spring

Binding Name (Optional)
bind-autoscaler

Add Parameters (Optional)
Name Value

BACK CREATE
```

d. Under Instance Name, enter a name for the instance.
e. (Optional) Enter a binding name under Binding Name in all lowercase letters. For example, bind-autoscaler.
f. (Optional) Specify additional parameters under Add Parameters. For a list of supported configuration parameters, consult the documentation for the service.
g. Click Create.

5. To unbind your app from a service instance, locate the service instance in the Bound Services list and click the three-dot icon on the far right. Select Unbind from the dropdown menu.

Map or Unmap Routes

1. Click Routes.

2. The page displays the routes associated with the app. To add a new route, click Map a Route.
3. Enter the route and click Map.

4. To unmap a route, locate the route from the list and click the red x. Click Unmap in the pop-up to confirm.

Create Container-to-Container Networking Policies

Container networking policies enable app instances to communicate with each other directly. You can create container networking policies in the Networking tab.

To view and use the Networking tab, you must have either the network.write or network.admin UAA scope. If you do not see the Networking tab, request one of the above scopes from your PCF administrator.

1. Click Networking. The page displays any container networking policies associated with the app.

2. To add a new network policy, click Create Policy or Add a network policy.
3. In the Add Policy window, configure the following:

- For Org, select the org of the destination app.
- For Space, select the space of the destination app.
- For App, select the destination app.
- For Protocol, select TCP or UDP.
- For Ports, enter the ports at which to connect to the destination app. The allowed range is from 1 to 65535. You can specify a single port, such as 8080, or a range of ports, such as 8080-8090.

4. Click Save.

For more information about container-to-container networking in PCF, see Container-to-Container Networking.
1. Click **Logs** to view the logs for the app.

2. Click the play button to view a live version of the logs.

**View Tasks**

1. Click the **Tasks** tab within Apps Manager.

2. This page displays a table containing **Task ID**, **State**, **Start Time**, **Task Name**, and **Command**.

**Run a Task**

1. Click **Run Task** to create a task.

2. (Optional) Enter a **Task Name**.

3. Enter the **Task Command**.

4. Click **Run**.

**Enable Task Scheduling**

In the **Tasks** tab, click **Enable Scheduling** to bind the PCF Scheduler service to your app. For more about the PCF Scheduler, see [Scheduling Jobs](#).
Schedule a Task

1. Navigate to the Tasks tab.

2. Click Create Job to schedule a task.

3. Enter a Job Name.

4. Enter a Command.

5. Enter one or more Cron Expressions for your desired task schedule or schedules. See Schedule a Job for more information on cron expression syntax.

6. Click Create Job.

View Settings

Click the Settings tab. In this tab, you can do the following:
- Rename the app.
- View information about the buildpack(s), start command, and stack.
- Configure health checks.
- View or add Environment Variables associated with the app.
- Add metadata to the app.
- View the Application Security Groups (ASGs) associated with the app.
- Delete the app. When you click Delete App, you also have the option to delete the app’s routes.

### Configure Health Checks

To configure health checks for your app, do the following:

1. Click the Settings tab.
2. In the Health Check section, click the Type dropdown and select `port`, `http`, or `process`. Depending on which option you select, additional fields may appear.
3. Complete the fields that appear with your selection.
   - For `port`, optionally complete the Invocation timeout (seconds) field.
For **http**, complete the **Endpoint** field. Optionally, complete the **Timeout (seconds)** and **Invocation timeout (seconds)** fields.

For **process**, there are no additional fields.

4. Click **Update**.

For more information, see the Using Application Health Checks topic.

**View or Add Environment Variables**

To add a user-provided environment variable, do the following:

1. Click the **Settings** tab.
2. Click **Reveal User Provided Env Vars**.
3. Enter the **Name** and **Value** of the variable. Alternatively, enter your variable using the **Enter JSON** toggle.
4. Click **Save**.

To view all environment variables, click **Reveal Env Vars**.
Add Metadata

You can add metadata, including labels and annotations, to apps and spaces using Apps Manager. For more information about adding metadata to spaces using Apps Manager, see Managing Orgs and Spaces Using Apps Manager.

For more information about adding metadata to objects in PAS, see Using Metadata.

To add labels or annotations to an app using Apps Manager, do the following:

1. Click the Settings tab of the desired app.
2. For Labels, provide a name and value pair. Or, select Enter JSON to enter the label in JSON.
3. For Annotations, provide a name and value pair. Or, select Enter JSON to enter the annotation in JSON.
4. To add more labels or annotations, click + next to the Labels or Annotations field.
5. Click Update Metadata.

Manage a Service Instance

From the Services tab on the space page, you can bind or unbind apps, bind or unbind routes, view or change your service plan, manage service keys, and rename or delete your service instance.

For services that use on-demand brokers, the service broker will create, update, or delete the service instance in the background and notify you when it finishes.

Bind an App

Note: For services that use asynchronous bindings, Apps Manager displays the status of the service while the bind is still pending. Asynchronous bindings provide more flexibility for services that require additional time before returning a successful bind response.

1. From the space page Services tab, click the service instance you want to bind to an app.
2. Click Bind App. A popup appears.

3. In the Bind App popup, select the app you want to bind to your service instance.

   ![Bind App Popup]

4. (Optional) To attach parameters to the binding, click Show Advanced Options. Under Arbitrary Parameters, enter any additional service-specific configuration.

5. Click Bind.

### Unbind an App

1. From the space page Services tab, click the service instance you want to unbind from an app.
2. Locate the app under Bound Apps and click the red × on the right. An Unbind App popup appears.

3. Click Unbind to confirm.

Share Service Instance

From the Overview tab, you can share your service instance across spaces. You can also view and delete service instances.

To share your service instance, do the following:

1. From the Overview tab of your service instance, click Share Service Instance.
2. Select the spaces you want to share your service instance with.

3. Click Share.

**View or Change Your Service Plan**

1. From the space page Services tab, click the service instance you want to view or change the plan for.

2. Click Plan.
3. Review your current plan information.

4. To change your plan, select a new plan from the list and click Select This Plan or Upgrade Your Account.

Note: Not all services support upgrading. If your service does not support upgrading, the service plan page only displays the selected plan.

Rename or Delete Your Service Instance

1. From the space page Services tab, click the service instance you want to rename or delete.

2. Click Settings.

- To change the service instance name, enter the new name and click Update.
- To add configuration parameters to the service instance, enter the parameters in the Name and Value fields and then click Update. Alternatively, enter your configuration parameters using the Enter JSON toggle and then click Update.
- To delete the service instance, click Delete Service Instance.

Note: The service broker supports creating, updating, and deleting service instances asynchronously. When the service broker completes one of these operations, a status banner appears in Apps Manager.
Update Your User-Provided Service Instance

Follow the steps below to update an existing user-provided service instance. You can create a user-provided service instance from the Marketplace. For more information, see the User-Provided Service Instances topic.

1. Click Configuration. This tab only appears for user-provided service instances.

   ![Configuration Tab](image)

   **Configuration**

   **Credential Parameters** (Optional)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   **Syslog Drain Url** (Optional)

   ![Syslog Drain Url](image)

   **Route Service Url** (Optional)

   ![Route Service Url](image)

   

   ![Update Service Button](image)

   ![Cancel Button](image)

2. Enter your Credential Parameters, Syslog Drain Url, and Route Service Url, and click Update Service.

Manage Service Keys

On the space page, click Services, then click the service instance that you want to manage service keys for. This directs you to the service instance Overview page, where you can generate a new service key, get the credentials for a service key, and delete a service key.

![Service Key Management](image)

Generate a Service Key

Follow the steps below to generate a service key.
1. In the Service Key Credentials section, click Create Service Key.

2. Edit the Service Key Name.

3. (Optional) Click Show Advanced Options. Under Arbitrary Parameters, enter any additional service-specific configuration in the Name and Value fields.

4. Click Create to generate the service key.

View Credentials for a Service Key

Follow the steps below to view the credentials for a service key.

1. To view the credentials for a particular service instance, click the service instance name under Service Key Credentials. The JSON object containing the credentials appears.

2. Click Close.

Delete Service Key

To delete a service key, click the red x next to the service instance name.
Manage Route Services

For more information about route services, see the Route Services topic.

You can bind a new service instance to a route when creating the instance in the Marketplace, or you can manage route services for an existing service instance on the service instance page.

Bind a New Service Instance to a Route

Follow the steps below to bind a new service instance to a route.

1. Select the service from the Marketplace.

2. Under Bind to Route, either bind the service instance to an existing route or click Create Route to create a new custom route.

   Note: You must choose a Marketplace service compatible with route services for the Bind to Route field to appear.

3. Complete the remaining fields and click Add to create the service instance.

Bind an Existing Service Instance to a Route

Follow the steps below to bind an existing service instance to a route.

1. On the space page, click Services.

2. Click the service instance that you want to manage route services for.

   Note: If the service is not compatible with route services, the text “This service does not support route binding” appears under Bound Routes.

3. To bind the service instance to a route, click Bind Route.
4. Select an existing route under Select a route to bind or enter a new route under Create Custom Route.

5. Click Bind.

To unbind a route from a service instance, click the red x next to the name of the route under Bound Routes.
Scaling an App Using App Autoscaler

This topic describes how Pivotal Cloud Foundry (PCF) users with the Space Developer role can set up and configure the App Autoscaler service in the Apps Manager UI to automatically scale apps based on rules that they set.

You can use the App Autoscaler command-line interface (CLI) plugin to configure App Autoscaler rules from the command line.

NOTE: Space Managers, Space Auditors, and all Org roles do not have permission to use App Autoscaler. For help managing user roles, see Managing User Accounts and Permissions Using the Apps Manager.

App Autoscaler Overview

App Autoscaler is a marketplace service that helps control the cost of running apps while maintaining app performance.

To balance app performance and cost, Space Developers can use App Autoscaler to do the following:

- Configure rules that adjust instance counts based on metrics thresholds, such as CPU Usage
- Modify the maximum and minimum number of instances for an app, either manually or following a schedule

Breaking Change: App Autoscaler relies on API endpoints from Loggregator’s Log Cache. If you disable Log Cache, App Autoscaler will fail. For more information about Log Cache, see Loggregator Introduces Log Cache.

Set Up App Autoscaler

To use App Autoscaler, you must create an instance of the App Autoscaler service and bind it to any app you want to autoscale. You can do this using either the Apps Manager or from the Cloud Foundry Command Line Interface (cf CLI):

- Apps Manager:
  1. Create an instance of the service.
  2. Bind the service to an app.

- cf CLI:
  1. Create an instance of the service.
  2. Bind the service to an app.

Follow these steps to enable App Autoscaler for the app:

1. In Apps Manager, select an app from the space in which you created the App Autoscaler service.
2. Under Processes and Instances, enable Autoscaling.
Click Manage Autoscaling to configure instance limits, scaling rules, and scheduled limit changes for the app.

### Manual Scaling Overrides Autoscaler

If you manually scale an app bound to an App Autoscaler service instance, the App Autoscaler instance automatically unbinds from that app, and the app scales to the manual setting.

### Configure Autoscaling for an App

App Autoscaler keeps instance counts within an allowable range defined by minimum and maximum values, or *instance limits*.

Follow the procedures in the sections below to set any of the following:

- Instance Limits
- Scaling Rules
- Scheduled Limit Changes
Instance Limits

Follow these steps to manually modify instance limits:

1. In Apps Manager, navigate to the Overview page for your app and click Manage Autoscaling under Processes and Instances.

2. In the INSTANCE LIMITS section, set values for Minimum and Maximum.

3. Click APPLY CHANGES.

Scaling Rules

To keep your apps available without wasting resources, App Autoscaler increments or decrements instance counts based on how current metrics compare with configurable minimum and maximum thresholds.

How App Autoscaler Determines When to Scale

App Autoscaler scales apps as follows:

- Increment by one instance when any metric exceeds the maximum threshold specified
- Decrement by one instance only when all metrics fall below the minimum threshold specified

Scaling Rule Metrics

The table below lists the metrics that you can base App Autoscaler rules on:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Utilization</td>
<td>Average CPU percentage for all instances of the app.</td>
</tr>
<tr>
<td>Container Memory Utilization</td>
<td>Average memory percentage for all instances of the app.</td>
</tr>
<tr>
<td>HTTP Throughput</td>
<td>Total HTTP requests per second (divided by the total number of app instances).</td>
</tr>
<tr>
<td>HTTP Latency</td>
<td>Average latency of apps response to HTTP requests. This does not include Gorouter processing time or other network latency. Average is calculated on the middle 99% or middle 95% of all HTTP requests.</td>
</tr>
<tr>
<td>RabbitMQ Depth</td>
<td>The queue length of the specified queue.</td>
</tr>
<tr>
<td>Custom</td>
<td>Users can configure their apps to emit custom metrics out of the Loggregator Firehose. For steps on how to set up your apps to emit custom metrics, refer to the Metrics Forwarder Documentation.</td>
</tr>
<tr>
<td>Compare</td>
<td>One custom metric divided by another. When you add a rule, the Metric field is the dividend and the Comparison Metric field is the divisor.</td>
</tr>
</tbody>
</table>

Add a Scaling Rule

1. In the Manage Autoscaling pane, click EDIT next to SCALING RULES. The Edit Scaling Rules pane appears.
2. Click ADD RULE.

3. In the Select type dropdown, select the metric for the new scaling rule.

4. Set the minimum and maximum thresholds for the metric using the table in the Scaling Rule Metrics section above as a guide.

5. Select or fill in any other fields that appear under the threshold fields:
   - If you are adding an HTTP Latency rule, configure Percent of traffic to apply.
   - If you are adding a RabbitMQ depth rule, provide the name of the queue to measure.
   - If you are adding a Custom rule, enter your custom Metric.
   - If you are adding a Compare rule, enter values in the Metric and Comparison Metric fields.

6. Click SAVE.

Delete a Scaling Rule

1. Click the × icon next to the rule you want to delete.

2. Click SAVE.

Scheduled Limit Changes

Because app demand often follows a weekly, daily, or hourly schedule, you can schedule App Autoscaler to change the allowable instance range to track expected surges or quiet periods.

Create or Modify a Scheduled Limit Change

1. Click EDIT next to SCHEDULED LIMITS.

2. Click ADD NEW to add a new scheduled limit or select an existing entry to modify by clicking EDIT next to the entry.

3. Edit the following values:
   - Date and Time (local): Set the date and time of the change.
   - Repeat (Optional): Set the day of the week for which you want to repeat the change.
   - Min and Max: Set the allowable range within which App Autoscaler can change the instance count for an app.
4. Click SAVE.

To delete an existing entry, click the x icon next to the entry you want to delete.

Example: Scale Down for the Weekend

To schedule an app to scale down for a weekend, you can enter two rules as follows:

1. Scale down to a single instance on Friday evening:
   - Date and Time (local): Dec 2, 2018 and 7:00 PM
   - Repeat (Optional): Fr
   - Min and Max: 1 and 1

2. Increase instances to between 3 and 5 on Monday morning:
   - Date and Time (local): Dec 5, 2018 and 7:00 AM
   - Repeat (Optional): M
   - Min and Max: 3 and 5

App Autoscaler Events and Notifications

App Autoscaler logs all autoscaling events.

View Event History

To view all autoscaling events in the past 24 hours, click VIEW MORE in the Event History section of the Manage Autoscaling pane.

Manage App Autoscaler Notifications

Autoscaler emails or texts its event notifications to all space users by default.
Follow these steps to subscribe or unsubscribe from autoscaling event notifications:

1. Navigate to the Manage Notifications page of PCF.

   Note: If installed, Notifications Management should be available at [https://notifications-ui.YOUR-SYSTEM-DOMAIN/preferences](https://notifications-ui.YOUR-SYSTEM-DOMAIN/preferences).

2. Choose which notifications you want to receive from App Autoscaler:

   ![Manage Notifications](image)

   You will receive email notifications from checked items:
   - Cloud Foundry Autoscaling Service
   - Scaling Down
   - Manual Scaling Detected
   - Maximum Instance Limit Reached
   - Quota Limit Reached
   - Scaling Up
Using the App Autoscaler CLI

This topic explains how to use the App Autoscaler command-line interface (CLI).

The App Autoscaler automatically scales Cloud Foundry apps in response to demand. The App Autoscaler CLI lets you control the App Autoscaler from your local command line by extending the Cloud Foundry command-line interface (cf CLI).

To use the App Autoscaler CLI, you must first install the App Autoscaler CLI plugin.

Install the App Autoscaler CLI Plugin

Before you can run App Autoscaler CLI commands on your local machine, you must install the App Autoscaler CLI plugin.

To install the App Autoscaler CLI plugin, download it from Pivotal Network.

**NOTE:** Ensure that you download the 2.0 release of the App Autoscaler CLI plugin. The App Autoscaler CLI is dependent on Pivotal Application Service 2.2.

View Apps

Run `cf autoscaling-apps` to view all the apps that are bound to an autoscaler service instance in a space, their instance limits, and whether or not they are enabled.

```
$ cf autoscaling-apps
```

Presenting autoscaler apps in org my-org / my-space autoscaling as user

<table>
<thead>
<tr>
<th>Name</th>
<th>Guid</th>
<th>Enabled</th>
<th>Min Instances</th>
<th>Max Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-app</td>
<td>guid</td>
<td>true</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>test-app-2</td>
<td>guid-2</td>
<td>false</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

Enable Autoscaling

Run `cf enable-autoscaling APP-NAME` to enable autoscaling on your app. Replace `APP-NAME` with the name of your app.

```
$ cf enable-autoscaling test-app-2
```

Enabled autoscaling for app test-app-2 for org my-org / my-space testing as admin

OK

Disable Autoscaling

Run `cf disable-autoscaling APP-NAME` to disable autoscaling on your app. Replace `APP-NAME` with the name of your app.

```
$ cf disable-autoscaling test-app
```

Disabled autoscaling for app test-app for org my-org / my-space testing as admin

OK

Update Instance Limits

Run `cf update-autoscaling-limits APP-NAME MIN-INSTANCE-LIMIT MAX-INSTANCE-LIMIT` to update the upper and lower app instance limits. The App Autoscaler will not attempt to scale beyond these limits. Replace `APP-NAME` with the name of your app. Replace `MIN-INSTANCE-LIMIT` with the minimum number of apps, and `MAX-INSTANCE-LIMIT` with the maximum number of apps.
View Rules

Run `cf autoscaling-rules APP-NAME` to view the rules that the App Autoscaler uses to determine when to scale your app. Replace `APP-NAME` with the name of your app.

Create a Rule


Replace the placeholders as follows:

- `APP-NAME` is the name of your app.
- `RULE-TYPE` is the type of your scaling rule.
- `MIN-THRESHOLD` is the minimum threshold for the metric.
- `MAX-THRESHOLD` is the maximum threshold for the metric.

You can use the following command options:

- `--metric`, `-m` is the metric for a Custom rule.
- `--comparison-metric`, `-c` is the comparison metric for a Compare rule.
- `--subtype`, `-s` is the rule subtype.

For example:

```
$ cf create-autoscaling-rule test-app http_latency 10 20 -s avg_99th
Created autoscaler rule for app test-app in org my-org / space my-space as user

<table>
<thead>
<tr>
<th>Guid</th>
<th>Type</th>
<th>Metric</th>
<th>Sub Type</th>
<th>Min Threshold</th>
<th>Max Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>guid-3</td>
<td>http</td>
<td>http_latency</td>
<td>avg_99th</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>
```

Valid Rule Types and Subtypes

For a list of valid types and subtypes, see the following:
Delete a Rule

Run `cf delete-autoscaling-rule APP-NAME RULE-GUID [--force]` to delete a single autoscaling rule. Replace `APP-NAME` with the name of your app, and replace `RULE-GUID` with the GUID.

```
$ cf delete-autoscaling-rule test-app guid-2
Really delete rule guid-2 for app test-app? [y/N]: y
Deleted rule guid-2 for autoscaler app test-app for org my-org / space my-space as admin
OK
```

Delete All Rules

Run `cf delete-autoscaling-rules APP-NAME [--force]` to delete all autoscaling rules. Replace `APP-NAME` with the name of your app.

```
$ cf delete-autoscaling-rules test-app
Really delete ALL rules for app test-app? [y/N]: y
Deleted rules for autoscaler app test-app for org my-org / space my-space as admin
OK
```

View Autoscaling Events

Run `cf autoscaling-events APP-NAME` to view recent events related to autoscaling for your app. Replace `APP-NAME` with the name of your app.

```
$ cf autoscaling-events test-app
Time  Description
2032-01-01T00:00:00Z  Scaled up from 3 to 4 instances. Current cpu of 20 is above upper threshold of 10.
```

View Autoscaler Scheduled Instance Limit Changes

Run `cf autoscaling-slcs APP-NAME` to view all scheduled instance limit changes. Replace `APP-NAME` with the name of your app.

For example:

```
$ cf autoscaling-slcs test-app
Guid  First Execution  Min Instances  Max Instances  Recurrence
guid-5  2018-06-12T22:00:00Z  0  1  Mo,Tu,We,Th,Fr
```
Create Autoscaler Scheduled Instance Limit Change

Run `create-autoscaling-slc APP-NAME DATE-TIME MIN-INSTANCES MAX-INSTANCES [--recurrence RECURRENCE]` to create a new scheduled instance limit change.

Replace the placeholders as follows:

- **APP-NAME** is the name of your app.
- **DATE-TIME** is the date and time of the change.
- **MIN-INSTANCES** and **MAX-INSTANCES** are the minimum and maximum values of the range within which App Autoscaler can change the instance count for an app.
- **RECURRENCE** (optional) is the day of the week for which you want to repeat the change.

For example:

```
cf create-autoscaling-slc test-app 2018-06-14T15:00:00Z 1 2 --recurrence Sa
```

Created scheduled autoscaler instance limit change for app test-app in org my-org / space my-space as user

OK

<table>
<thead>
<tr>
<th>guid</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>First Execution</td>
</tr>
<tr>
<td>guid-6</td>
</tr>
</tbody>
</table>

Delete Autoscaler Scheduled Instance Limit Change

Run `delete-autoscaling-slc APP-NAME SLC-GUID [--force]` to delete a scheduled instance limit change. Replace **APP-NAME** with the name of your app and **SLC-GUID** with the GUID of your scheduled instance limit change.

For example:

```
cf delete-autoscaling-slc test-app slc-guid
```

Really delete scheduled limit change slc-guid for app test-app? [yN]: y

Deleted scheduled limit change slc-guid for app test-app in org my-org / space my-space as user

OK

Configure with a Manifest

Run `configure-autoscaling APP-NAME MANIFEST-FILE-PATH` to use a service manifest to configure your rules, add instance limits, and set scheduled limit changes at the same time. Replace **APP-NAME** with the name of your app, and replace **MANIFEST-FILE-PATH** with the path and name of your Autoscaler manifest.

An example manifest:

```yaml
---
instance_limits:
  min: 1
  max: 2
rules:
- rule_type: "http_latency"
  rule_sub_type: "avg_99th"
  threshold:
    min: 10
    max: 20
scheduled_limit_changes:
- recurrence: 10
  executes_at: "2032-01-01T00:00:00Z"
  instance_limits:
    min: 10
    max: 20
```

```
cf configure-autoscaling test-app autoscaler-manifest.yml
```

Setting autoscaler settings for app test-app for org my-org / space my-space as user

OK
A **rules** block must be present in your Autoscaler manifest. If your app does not require any rules changes, include an empty block:

```yaml
---
instance_limits:
  min: 1
  max: 1
rules: []
scheduled_limit_changes:
  - recurrence: 365
    executes_at: "2032-01-01T00:00:00Z"
  instance_limits:
  min: 0
  max: 0
---
```

A **scheduled_limit_changes** block must be present in your Autoscaler manifest. If your app does not require any scheduled instance limit changes, include an empty block:

```yaml
---
instance_limits:
  min: 1
  max: 2
rules:
  - rule_type: "http_latency"
    rule_sub_type: "avg_99th"
    threshold:
      min: 10
      max: 20
    scheduled_limit_changes: []
---
```

### App Autoscaler CLI Known Issues

The App Autoscaler CLI has the following known issues:

- The CLI returns an error message if you have more than one instance of the App Autoscaler service running in the same space.
  - To prevent this error, delete all but one App Autoscaler service instance from any space that the App Autoscaler service runs in.

- The CLI may output odd characters in Windows shells that do not support text color.
  - To prevent this error, run `SET CF_COLOR=false` in your Windows shell pane before you run App Autoscaler CLI commands.
  - Note that some Windows shells do not support the `CF_COLOR` setting.
Viewing ASGs in Apps Manager

Page last updated:

About ASGs

Application Security Groups (ASGs) are a collections of egress rules that specify the protocols, ports, and IP address ranges where app or task instances send traffic. The platform sets up rules to filter and log outbound network traffic from app and task instances. ASGs apply to both buildpack-based and Docker-based apps and tasks.

When apps or tasks begin staging, they need traffic rules permissive enough to allow them to pull resources from the network. After an app or task is running, the traffic rules can be more restrictive and secure. To distinguish between these two security requirements, administrators can define one ASG for app and task staging, and another for app and task runtime. For more information about staging and running apps, see Application Container Lifecycle.

To provide granular control when securing a deployment, an administrator can assign ASGs to apply to all app and task instances for the entire deployment, or assign ASGs to spaces to apply only to apps and tasks in a particular space.

Only admin users can create and modify ASGs. For information about creating and configuring ASGs, see Application Security Groups.

Displaying ASGs for a Space

To view the ASGs associated with a space, perform the following steps.

1. Log in to Apps Manager.
2. From the Home page, select the Org that contains the space you want to view.
3. Select the Space you want to view.
4. Click on the Settings tab.
5. In the Security Groups section, Apps Manager displays ASGs associated with the selected space.
6. Click on an ASG to expand its egress rules.
Configuring Spring Boot Actuator Endpoints for Apps Manager

Page last updated:

The Apps Manager UI supports several production-ready endpoints from Spring Boot Actuator. This topic describes the Actuator endpoints and how you can configure your app to display data from the endpoints in Apps Manager.

For more information about Spring Boot Actuator, see the [Spring Boot Actuator documentation](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html).

⚠️ Note: This feature requires Spring Boot v1.5 or later.

Overview

The Apps Manager integration with Spring Boot does not use the standard Spring Boot Actuators. Instead, it uses a specific set of actuators that are secured using the Space Developer role for the space that the application runs in. Authentication and authorization are automatically delegated to the Cloud Controller and the User Account and Authentication server without any configuration from the user.

By default, actuators are secure and cannot be accessed without explicit configuration by the user, even if Spring Security is not included. This allows users to take advantage of the Spring Boot Apps Manager integration without accidentally exposing their actuators without security.

Actuator Endpoints

The table below describes the Spring Boot Actuator endpoints supported by Apps Manager. To integrate these endpoints with Apps Manager, you must first [Activate Spring Boot Actuator for Your App](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html).

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>About</th>
</tr>
</thead>
</table>
| /info    | **Description:** Exposes details about app environment, git, and build. To send build and Git information to this endpoint, see [Configure the Info Actuator](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html).
  
  **How to use in Apps Man:** See [View Build and Git Information for Your App](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |
| /health  | **Description:** Shows health status or detailed health information over a secure connection. Spring Boot Actuator includes the auto-configured health indicators specified in the [Auto-configured HealthIndicators](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html) section of the Spring Boot documentation. If you want to write custom health indicators, see the [Writing custom HealthIndicators](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html) section of the Spring Boot documentation.
  
  **How to use in Apps Man:** See [View App Health](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |
| /loggers | **Description:** Lists and allows modification of the levels of the loggers in an app.
  
  **How to use in Apps Man:** See [Manage Log Levels](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |
| /dump    | **Description:** Generates a thread dump.
  
  **How to use in Apps Man:** See [View Dump](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |
| /trace   | **Description:** Displays trace information from your app for each of the last 100 HTTP requests. For more information, see the [Tracing](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html) section of the Spring Boot documentation.
  
  **How to use in Apps Man:** See [View Request Traces](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |
| /heapdump | **Description:** Generates a heap dump and provides a compressed file containing the results.
  
  **How to use in Apps Man:** See [Download Heap Dump](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |
| /mappings | **Description:** Displays the endpoints an app serves and other related details.
  
  **How to use in Apps Man:** See [View Mappings](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator-overview.html). |

Activate Spring Boot Actuator for Your App

© Copyright Pivotal Software Inc, 2013-2019
You must add a `spring-boot-starter-actuator` dependency to your app project for the production-ready HTTP endpoints to return values. For more information, see the Enabling production-ready features section of the Spring Boot documentation.

1. Follow the instructions below that correspond to your project type.

   - **Maven**: If you use Maven, add the following to your project:

     ```xml
     <dependencies>
     <dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-actuator</artifactId>
     </dependency>
     </dependencies>
     ```

   - **Gradle**: If you use Gradle, add the following to your project:

     ```gradle
     dependencies {
     compile('org.springframework.boot:spring-boot-starter-actuator')
     }
     ```

2. If you use self-signed certificates in your PCF deployment for UAA or the Cloud Controller, specify in your `application.properties` file to skip SSL validation:

   ```properties
   management.cloudfoundry.skip-ssl-validation=true
   ```

   See Cloud Foundry support in the Spring Boot Actuator documentation for more information.

**Configure the Info Actuator**

The `/info` endpoint provides information about the project build for your app, as well as its git details.

**Add Build Information**

To add build information to the `/info` endpoint, follow the instructions below that correspond to your project type.

**Maven**

Add the following to your app project:

```xml
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>1.4.2.RELEASE</version>
      <executions>
        <execution>
          <goals>
            <goal>build-info</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

**Gradle**

Add the following to your app project:

```gradle
springBoot {
  buildInfo()
}
```
Add Git Information

To add git information to the `/info` endpoint, follow these instructions:

1. Add the following property to your `application.properties` file:

```
management.info.git.mode=full
```

2. Follow the instructions below that correspond to your project type.

Maven

Add the following plugin to your project:

```
<build>
  <plugins>
    <plugin>
      <groupId>pl.project13.maven</groupId>
      <artifactId>git-commit-id-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

Gradle

Add the following plugin to your project:

```
plugins {
  id "com.gorylenko.gradle-git-properties" version "1.4.17"
}
```
Using Spring Boot Actuators with Apps Manager

This document describes how to view and manage app information from Spring Boot Actuator in Apps Manager.

Prerequisites

The Apps Manager integration with Spring Boot Actuator requires the following:

- A PCF user with the `SpaceDeveloper` role. See App Space Roles.
- Spring Boot v1.5 or later.
- Completing the procedures in Configure Spring Boot Actuator Endpoints for Apps Manager.

After your configure your app, Apps Manager displays the Spring Boot logo next to the name of your app on the app page:

![App Overview](image)

View Build and Git Information for Your App

To view the data that your app sends to its `/info` Actuator endpoint, select the Settings tab:
In the upper right of the app page, Apps Manager also displays the SHA of your app code repository from the latest build:
To view the health-check data that your app sends to its `/health` Actuator endpoints, select the **Overview** tab and click an instance under the **Instances** section:

<table>
<thead>
<tr>
<th>#</th>
<th>App Health</th>
<th>CPU</th>
<th>Memory</th>
<th>Disk</th>
<th>Uptime</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>Up</strong></td>
<td>0%</td>
<td>209.97 MB</td>
<td>137.75 MB</td>
<td>1 min</td>
</tr>
</tbody>
</table>

**View Thread Dump**

To trigger and view a thread dump from your app to its `/dump` Actuator endpoint, select the **Threads** tab and click **Refresh**.

You can click each thread to expand and view its details. You can also modify which threads appear on the page using the **Instance** and **Show** drop-down menus.

**View Request Traces**

To retrieve and view tracing information from the `/trace` Actuator of your app, select the **Trace** tab and click **Refresh**.
This page displays the last 100 requests from your app. You can click each individual request to expand and view its trace details. You can modify which requests appear on the page using the Instance drop-down menu.

By default, the Trace tab does not show requests and responses from Apps Manager polling app instances for data. To include these requests, clear the Hide Pivotal Apps Manager Requests checkbox next to the Instance drop-down menu.

**Download Heap Dump**

To trigger and view a heap dump from your app to its /heapdump endpoint, select the settings drop-down menu for an instance of your app and click Heap Dump. This downloads a .zip file.
View Mappings

To view a collated list of the endpoints an app serves, select the Settings tab and click View Mappings.

Manage Log Levels

Spring Boot apps include loggers for many provided and user components of the app. You can set the log level for each logger in Apps Manager.

To view the Configure Logging Levels screen, select the Logs tab and click Configure Logging Levels.
Apps Manager displays the default log level for each logger in gray.

You can modify the log level for a logger by clicking the desired level in the logger row, as shown in the image below. Whenever you set a log level, the following happens:

- The log level displays in blue to indicate that it is user-configured.
- Each child namespace of the logger inherits the log level.

**Note:** You can manually set any of the child loggers to override this inheritance.
All of the loggers with user-configured logging levels float to the top of the list.

You can reset log levels by clicking the white dot displayed on the current log level.

You can also filter which loggers you see using the Filter Loggers textbox.
Troubleshoot Spring Boot Actuator Integration

This section describes how to troubleshoot common issues with the integration of Apps Manager and Spring Boot Actuator.

/cloudfoundryapplication Failed Request

Symptom
You see the following failed request message in your app logs:

```
Could not find resource for relative : /cloudfoundryapplication of full path: http://example.com/cloudfoundryapplication
```

Explanation
 Apps Manager uses the `/cloudfoundryapplication` endpoint as the root for Spring Boot Actuator integrations. It calls this endpoint for an app when you view the app in the Apps Manager UI, regardless of whether you have configured Spring Boot Actuator endpoints for Apps Manager.

Solution
If you are not using the Spring Boot Actuator integrations for Apps Manager, you can ignore this failed request message.
Configuring Multi-Foundation Support in Apps Manager

This topic describes how to configure multi-foundation support in Apps Manager.

Overview

Configuring multi-foundation support in Apps Manager allows you to search, view, and manage orgs, spaces, apps, and service instances across multiple foundations from a single interface.

Configure Multi-Foundation Support

This section describes the procedures for configuring multi-foundation support in Apps Manager.

To configure multi-foundation support in Apps Manager, do the following:

1. Configure PAS. See Configure PAS.
2. Add redirects. See Add Redirects in UAA.
4. (Optional) Configure the SAML Identity Provider. See (Optional) Configure the SAML Identity Provider.

Configure PAS

This section describes how to configure Pivotal Application Service (PAS) on a foundation to enable multi-foundation support in Apps Manager.

To configure multi-foundation support in Apps Manager, do the following:

1. In the PAS tile of one of your foundations, select Apps Manager. The following image shows the Apps Manager pane.
2. For Multi-foundation Configuration (BETA), enter a JSON object for each additional foundation that you want to manage. Use the following format for the JSON object:

```json
{
  "FOUNDATION_NAME": {
    "ccUrl": "https://api.FOUNDATION_SYSTEM_DOMAIN.com",
    ...
  }
}
```
Where:
- **FOUNDATION_NAME** is a name for the foundation.
- **FOUNDATION_SYSTEM_DOMAIN** is the system domain of the foundation. The system domain is listed in the PAS tile Domains pane.
- **metricsUrl** is an optional field. If PCF Metrics is installed on the foundation, provide the PCF Metrics URL.

Add Redirections in UAA

The Apps Manager UAA clients on each foundation you want to manage must include a redirect to the foundation on which you enabled multi-foundation support. This allows Apps Manager to redirect to the other foundations and back when authenticating with UAA. For more information about UAA clients and configuring valid redirect URIs, see [Client.redirect-uri](#) in UAA Concepts.

To add redirects, do the following:

1. Install the UAA CLI. See [CloudFoundry UAA Command Line Client](#) in GitHub.

2. Record the UAA admin client credentials password for a foundation you want to manage with Apps Manager:
   - a. In the PAS tile Credentials tab, for Admin Client Credentials, click Link to Credentials.
   - b. Record the value in the password field.

3. Use the following command to target UAA on the foundation:
   ```
   uaac target uaa.FOUNDATION_SYSTEM_DOMAIN.com
   ```
   Where **FOUNDATION_SYSTEM_DOMAIN** is the system domain of the foundation.

4. Use the following command to generate a token for the UAA admin client:
   ```
   uaac token client get admin -s ADMIN_CLIENT_PASSWORD
   ```
   Where **ADMIN_CLIENT_PASSWORD** is the Admin Client Credentials password you recorded in step 2.

5. Record the UAA redirect URIs. Use the following command to view the redirect URIs on the UAA apps_manager_js client:
   ```
   uaac client get apps_manager_js
   ```

6. Add the URI of the foundation on which you enabled multi-foundation support to the apps_manager_js client:
   ```
   uaac client update apps_manager_js --redirect_uri 'PREVIOUS_REDIRECT_URIS,https://apps.FOUNDATION_SYSTEM_DOMAIN.com/**'
   ```
   Where:
   - **PREVIOUS_REDIRECT_URIS** is one or more redirect URIs that you recorded in the previous step. Include the previous redirect URIs in the command to ensure they are not overwritten.
   - **FOUNDATION_SYSTEM_DOMAIN** is the system domain of foundation on which you enabled multi-foundation support.

7. Repeat steps 2 through 6 for each foundation you want to manage with Apps Manager.

Add Trusted Certificate Authorities

Apps Manager must be able to validate the CAs used by all foundations.

If any foundation uses a certificate for TLS termination that is not signed by a globally-trusted CA, add the CA that signed the TLS certificate as a trusted CA on the foundation on which you enabled multi-foundation support.
For more information, see Setting Trusted Certificates.

(Optional) Configure the SAML Identity Provider

Pivotal recommends that all foundations use the same external SAML identity provider. This allows Apps Manager to automatically authenticate with the identity provider.

To configure all foundations to use the same SAML identity provider, do the following for each foundation:

1. Navigate to the PAS tile UAA pane.
2. Verify that the foundation uses the correct provider for Provider Name.

For more information on UAA and identity providers, see Identity Providers in UAA.

For more information on how to setup identity providers in UAA, see Adding Existing SAML or LDAP Users to a PCF Deployment.
Using the Cloud Foundry Command Line Interface (cf CLI)

This guide explains the Cloud Foundry Command Line Interface (cf CLI), a tool you use to deploy and manage your applications.

Contents in this section:

- Installing the cf CLI
- Getting Started with the cf CLI
- Using the cf CLI with an HTTP Proxy Server
- Using the cf CLI with a Self-Signed Certificate
- Using cf CLI Plugins
- Developing cf CLI Plugins
- Cloud Foundry CLI Reference Guide
- Using Experimental cf CLI Commands
Installing the cf CLI

Page last updated:

This topic describes how to install the Cloud Foundry Command Line Interface (cf CLI). Follow the instructions below for your operating system. If you previously used the cf CLI Ruby gem, uninstall this gem first.

You can install the cf CLI with a package manager, an installer, or a compressed binary.

Note: For use with Pivotal Cloud Foundry v1.10, the recommended minimum version is cf CLI v6.23 or later.

Use a Package Manager

Mac OS X Installation

For Mac OS X, perform the following steps to install the cf CLI with Homebrew:

1. Tap the Cloud Foundry formula repository:

   $ brew tap cloudfoundry/tap

2. Install the cf CLI:

   $ brew install cf-cli

Linux Installation

For Debian and Ubuntu-based Linux distributions, perform the following steps:

1. Add the Cloud Foundry Foundation public key and package repository to your system:

   $ wget -q -O /etc/apt/sources.list.d/cloudfoundry-cli.list https://packages.cloudfoundry.org/debian/cli.cloudfoundry.org.list

   $ echo "deb https://packages.cloudfoundry.org/debian stable main" | sudo tee /etc/apt/sources.list.d/cloudfoundry-cli.list

2. Update your local package index:

   $ sudo apt-get update

3. Install the cf CLI:

   $ sudo apt-get install cf-cli

For Enterprise Linux and Fedora systems (RHEL6/CentOS6 and up), perform the following steps:

1. Configure the Cloud Foundry Foundation package repository:

   $ sudo wget -O /etc/yum.repos.d/cloudfoundry-cli.repo https://packages.cloudfoundry.org/fedora/cloudfoundry-cli.repo

2. Install the cf CLI, which also downloads and adds the public key to your system:

   $ sudo yum install cf-cli
Use an Installer

Follow the instructions for your operating system below.

Windows Installation

You can run cf CLI in either the Windows Subsystem for Linux (WSL), also known as Bash on Windows, or in the Windows command line.

To use WSL, follow the Linux Installation instructions.

To use the cf CLI installer for the Windows command line, perform the following steps:

1. Download the Windows installer.
2. Unpack the zip file.
3. Right click on the ‘cf_installer’ executable and select “Run as Administrator”
4. When prompted, click Install, then Finish.
5. To verify your installation, open a command prompt and type `cf`. If your installation was successful, the cf CLI help listing appears. You may need to restart the command prompt to see the cf CLI help listing appear.

Mac OS X Installation

To use the cf CLI installer for Mac OS X, perform the following steps:

1. Download the OS X installer.
2. Open the .pkg file.
3. In the installer wizard, click Continue.
4. Select an install destination and click Continue.
5. When prompted, click Install.
6. To verify your installation, open a terminal window and type `cf`. If your installation was successful, the cf CLI help listing appears.

Linux Installation

To use the cf CLI installer for Linux, perform the following steps:

1. Download the Linux installer for your Debian/Ubuntu or Red Hat system.
2. Install using your system’s package manager. Note these commands may require `sudo`.
   - For Debian/Ubuntu, run the following command:
     ```
     dpkg -i path/to/cf-cli-*.deb &
     apt-get install -f
     ```
   - For Red Hat, run the following command:
     ```
     rpm -i path/to/cf-cli-*.rpm
     ```
3. To verify your installation, open a terminal window and type `cf`. If your installation was successful, the cf CLI help listing appears.

Use a Compressed Binary

Download the compressed binary for Mac OS X, Windows, or Linux from the cf CLI GitHub repository and install it on your system.
The specific procedures vary by operating system, but the following example illustrates downloading and installing the binary on Mac OS X:

1. Download and extract the Mac OS X binary:

   ```bash
   ```

2. Move it to `/usr/local/bin`, or another location in your `$PATH`:

   ```bash
   mv cf /usr/local/bin
   ```

3. Confirm your cf CLI version:

   ```bash
   cf --version
   ```

Next Steps

See [Getting Started with cf CLI](#) for more information about how to use the cf CLI.

We recommend that you review our [CLI releases](#) to learn when updates are released, and download a new binary or a new installer when you want to update to the latest version.

Uninstall the cf CLI

**Package Manager**

If you previously installed the cf CLI with a package manager, follow the instructions specific to your package manager to uninstall the cf CLI.

The specific procedures vary by package manager, but the following example illustrates uninstalling the cf CLI with Homebrew:

```bash
brew uninstall cf-cli
```

**Installer**

If you previously installed the cf CLI with an installer, perform the instructions specific to your operating system to uninstall the cf CLI:

- For Mac OS, delete the binary `/usr/local/bin/cf`, and the directory `/usr/local/share/doc/cf-cli`.
- For Windows, navigate to the Control Panel, click Programs and Features, select Cloud Foundry CLI VERSION and click Uninstall.

**Binary**

If you previously installed a cf CLI binary, remove the binary from where you copied it.

cf CLI v6

To uninstall, run

```bash
gem uninstall cf
```

**Note:** To ensure that your Ruby environment manager registers the change, close and reopen your terminal.
Getting Started with the cf CLI

Page last updated:

This topic describes configuring and getting started with the Cloud Foundry Command Line Interface (cf CLI). This page assumes you have the latest version of the cf CLI. See the Installing the Cloud Foundry Command Line Interface topic for installation instructions.

Localize

The cf CLI translates terminal output into the language that you select. The default language is en-US. The cf CLI supports the following languages:

- Chinese (simplified): zh-Hans
- Chinese (traditional): zh-Hant
- English: en-US
- French: fr-FR
- German: de-DE
- Italian: it-IT
- Japanese: ja-JP
- Korean: ko-KR
- Portuguese (Brazil): pt-BR
- Spanish: es-ES

Use `cf config` to set the language. To set the language with `cf config`, use the syntax: `cf config --locale YOUR_LANGUAGE`.

For example, to set the language to Portuguese and confirm the change by running `cf help`:

```
$ cf config --locale pt-BR
$ cf help
NAME: cf
- Uma ferramenta de linha de comando para interagir com Cloud Foundry
USAGE: cf [opções globais] comando [argumentos...][opções de comando]
```

Note: Localization with `cf config --locale` affects only messages that the cf CLI generates.

Login

Use `cf login` to log in to PAS. The `cf login` command uses the following syntax to specify a target API endpoint, an org (organization), and a space:

```
$ cf login [a_API_URL] [-u USERNAME] [-p PASSWORD] [-o ORG] [-s SPACE]
```

- **API_URL**: This is your API endpoint, the URL of the Cloud Controller in your PAS instance.
- **USERNAME**: Your username.
- **PASSWORD**: Your password. Use of the `-p` option is discouraged as it may record your password in your shell history.
- **ORG**: The org where you want to deploy your apps.
- **SPACE**: The space in the org where you want to deploy your apps.

The cf CLI prompts for credentials as needed. If you are a member of multiple orgs or spaces, `cf login` prompts you for which ones to log into. Otherwise it targets your org and space automatically.
 Alternately, you can write a script to log in and set your target using the non-interactive `cf api`, `cf auth`, and `cf target` commands.

Upon successful login, the cf CLI saves a \texttt{config.json} file containing your API endpoint, org, space values, and access token. If you change these settings, the \texttt{config.json} file is updated accordingly.

By default, \texttt{config.json} is located in your \texttt{~/.cf} directory. The \texttt{CF_HOME} environment variable allows you to locate the \texttt{config.json} file wherever you like.

Users and Roles

The cf CLI includes commands that list users and assign roles in orgs and spaces. See the Orgs, Spaces, Roles, and Permissions topic.

Commands for Listing Users

These commands take an org or space as an argument:

- \texttt{cf org-users}
- \texttt{cf space-users}

For example, to list the users who are members of an org:

```
S cf org-users example-org
Getting users in org example-org as username@example.com...

ORG MANAGER
username@example.com

BILLING MANAGER
huey@example.com
dewey@example.com

ORG AUDITOR
louie@example.com
```

Commands for Managing Roles

These commands require PAS admin permissions and take username, org or space, and role as arguments:

- \texttt{cf set-org-role}
- \texttt{cf unset-org-role}
- \texttt{cf set-space-role}
- \texttt{cf unset-space-role}

Available roles are OrgManager, BillingManager, OrgAuditor, SpaceManager, SpaceDeveloper, and SpaceAuditor. For example, to grant the Org Manager role to
a user within an org:

```bash
$ cf set-org role: Huey@example.com example-org OrgManager
Assigning role OrgManager to user huey@example.com in org example-org as username@example.com...
OK
```

**Note:** If you are not a PAS admin, you see this message when you try to run these commands:

```
error code: 10003, message: You are not authorized to perform the requested action
```

Identical Usernames in Multiple Origins

If a username corresponds to multiple accounts from different user stores, such as both the internal UAA store and an external SAML or LDAP store, the `set` and `unset` role commands above return an error:

```
The user exists in multiple origins. Specify an origin for the requested user from: 'uaa', 'other'
```

To resolve this ambiguity, the operator can construct a `curl` command that uses the CF API to perform the desired role-management function. For an example, see the [PUT v2/organizations/:guid/auditors](#) API function.

**Push**

The `cf push` command pushes a new app or syncs changes to an existing app.

If you do not provide a hostname (also known as subdomain), `cf push` routes your app to a URL of the form `APPNAME.DOMAIN` based on the name of your app and your default domain. If you want to map a different route to your app, see the [Routes and Domains](#) topic for information about creating routes.

The `cf push` command supports many options that determine how and where the app instances are deployed. For details about the `cf push` command, see the [push](#) page in the Cloud Foundry CLI Reference Guide.

The following example pushes an app called `my-awesome-app` to the URL `http://my-awesome-app.example.com` and specifies the Ruby buildpack with the `-b` flag.

```
$ cf push my-awesome-app -b ruby_buildpack
Creating app my-awesome-app in org example-org / space development as username@example.com...
OK
Creating route my-awesome-app.example.com...
OK...
1 of 1 instances running
App started...

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: my-awesome-app.example.com
stack: cflinuxfs3
buildpack: ruby_buildpack

state since cpu memory disk details
#0 running 2016-06-08 04:44:07 PM 0.0% 0 of 1G 0 of 1G
```

For more information about available buildpacks, see the [Buildpacks](#) topic.

Always Provide an App Name to cf push

© Copyright Pivotal Software Inc, 2013-2019 1088 2.5
**How cf push Finds the App**

By default, `cf push` recursively pushes the contents of the current working directory. Alternatively, you can provide a path using either a manifest or a command line option.

- If the path is to a directory, `cf push` recursively pushes the contents of that directory instead of the current working directory.
- If the path is to a file, `cf push` pushes only that file.

**Note:** If you want to push more than a single file, but not the entire contents of a directory, consider using a `.cfignore` file to tell `cf push` what to exclude.

---

## User-Provided Service Instances

To create or update a user-provided service instance, you need to supply basic parameters. For example a database service might require a username, password, host, port, and database name.

The cf CLI has three ways of supplying these parameters to create or update an instance of a service: interactively, non-interactively, and in conjunction with third-party log management software as described in RFC 6587. When used with third-party logging, the cf CLI sends data formatted according to RFC 5424.

You create a service instance with `cf cups` and update one with `cf uups` as described below.

---

### The cf create-user-provided-service (cups) Command

Use `cf create-user-provided-service` (alias `cf cups`) creates a new service instance.

To supply service instance parameters interactively: Specify parameters in a comma-separated list after the `-p` flag. This example command-line session creates a service instance for a database service.

```
$ cf cups sql-service-instance -p "host=mysql.example.com", "port=1433", dname="mysql", username="admin", password="Pa55w0rd"
Creating user provided service sql-service-instance in org example-org / space development as username@example.com...------------------------------------------>
OK
```

To supply service instance parameters to `cf cups` non-interactively: Pass parameters and their values in as a JSON hash, bound by single quotes, after the `-p` tag. This example is a non-interactive version of the session above.

```
$ cf cups sql-service-instance -p '{"host":"mysql.example.com","port":1433,"dname":"mysql","username":"admin","password":"Pa55w0rd"}'
Creating user provided service sql-service-instance in org example-org / space development as username@example.com...------------------------------------------>
OK
```

To create a service instance that sends data to a third-party: Use the `-l` option followed by the external destination URL. This example creates a service
instance that sends log information to the syslog drain URL of a third-party log management service. For specific log service instructions, see the Service-Specific Instructions for Streaming Application Logs topic.

```bash
$ cf cups mylog -l syslog://logs4.example.com:25258
Creating user provided service mylog in org example-org / space development as username@example.com...
OK
```

After you create a user-provided service instance, you bind it to an app with `cf bind-service`\(^1\), unbind it with `cf unbind-service`\(^1\), rename it with `cf rename-service`\(^1\), and delete it with `cf delete-service`\(^1\).

### The cf update-user-provided-service (uups) Command

Use `cf update-user-provided-service`\(^2\) (alias `cf uups`) to update one or more of the parameters for an existing user-provided service instance. The `cf uups` command uses the same syntax as `cf cups` above to set parameter values. The `cf uups` command does not update any parameter values that you do not supply.

### cf CLI Return Codes

The `cf` CLI uses exit codes, which help with scripting and confirming that a command has run successfully. For example, after you run a `cf` CLI command, you can retrieve its return code by running `echo $?` (on Windows, `echo %ERRORLEVEL%`). If the return code is `0`, the command was successful.

### The cf help Command

The `cf help` command lists the `cf` CLI commands and a brief description of each. Passing the `-h` flag to any command lists detailed help, including any aliases. For example, to see detailed help for `cf delete`, run:

```bash
$ cf delete -h
NAME:
   delete - Delete an app

USAGE:
   cf delete APP_NAME [-f -r]

ALIAS:
   d

OPTIONS:
   -f Force deletion without confirmation
   -r Also delete any mapped routes
```
Using the cf CLI with a Proxy Server

If you have an HTTP or SOCKS5 proxy server on your network between a host running the cf CLI and your Cloud Foundry API endpoint, you must set `https_proxy` with the hostname or IP address of the proxy server.

The `https_proxy` environment variable holds the hostname or IP address of your proxy server.

`https_proxy` is a standard environment variable. Like any environment variable, the specific steps you use to set it depends on your operating system.

Format of `https_proxy`

`https_proxy` is set with hostname or IP address of the proxy server in URL format:

```
https_proxy=http://proxy.example.com
```

If the proxy server requires a user name and password, include the credentials:

```
https_proxy=http://username:password@proxy.example.com
```

If the proxy server uses a port other than 80, include the port number:

```
https_proxy=http://username:password@proxy.example.com:8080
```

If the proxy server is a SOCKS5 proxy, specify the SOCKS5 protocol in the URL:

```
https_proxy=socks5://socks_proxy.example.com
```

Using SOCKS5 with v3-ssh

The `cfv3-ssh` command supports SOCKS5 proxies. To specify the SOCKS5 proxy server, set the `ALL_PROXY` environment variable using the following format:

```
ALL_PROXY=socks5://socks_proxy.example.com
```

Note: `cfv3-ssh` does not work through a SOCKS5 proxy.

Setting `https_proxy` in Mac OS or Linux

Set the `https_proxy` environment variable using the command specific to your shell. For example, in bash, use the `export` command.

Example:

```
$ export https_proxy=http://my.proxyserver.com:8080
```

To make this change persistent, add the command to the appropriate profile file for the shell. For example, in bash, add a line like the following to your `.bash_profile` or `.bashrc` file:

```
https_proxy=http://username:password@hostname:port
export https_proxy
```

Setting `https_proxy` in Windows

1. Open the Start menu. Right-click Computer and select Properties.
2. In the left pane of the System window, click Advanced system settings.

3. In the System Properties window, select the Advanced tab, then click Environment Variables.

4. In the Environment Variables window, under User variables, click New.
5. In the Variable name field, input `https_proxy`. In the Variable value field, input your proxy server information.

6. Click OK.
Using the cf CLI with a Self-Signed Certificate

Page last updated:

This topic describes how developers can use the cf CLI to communicate securely with a Pivotal Cloud Foundry (PCF) deployment without specifying `--skip-ssl-validation` under the following circumstances:

- The deployment uses a self-signed certificate.
- The deployment uses a certificate that is signed by a self-signed certificate authority (CA), or a certificate signed by a certificate that's signed by a self-signed CA.

Before following the procedure below, the developer must obtain either the self-signed certificate or the intermediate and CA certificate(s) used to sign the deployment's certificate. The developer can obtain these certificates from the PCF operator.

Install the Certificate on Local Machines

The certificates that developers must insert into their local truststore vary depending on the configuration of the deployment.

- If the deployment uses a self-signed certificate, the developer must insert the self-signed certificate into their local truststore.
- If the deployment uses a certificate that is signed by a self-signed certificate authority (CA), or a certificate signed by a certificate that's signed by a self-signed CA, the developer must insert the self-signed certificate and any intermediate certificates into their local truststore.

Installing the Certificate on Mac OS X

Enter the following command to place a certificate file `server.crt` into your local truststore:

```
$ sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain server.crt
```

Installing the Certificate on Linux

Perform the following steps specific to your distribution to place the certificate file `server.crt` into your truststore:

- Debian/Ubuntu/Gentoo:
  
  ```
  $ cat server.crt >> /etc/ssl/certs/ca-certificates.crt
  ```

- Fedora/RHEL:
  
  ```
  $ cat server.crt >> /etc/pki/tls/certs/ca-bundle.crt
  ```

The above example will set certificate permanently on your machine across all users and requires sudo permissions. You can also run the following command to set certificate in your current terminal/script:

```
$ export SSL_CERT_FILE=/path/to/server.crt
```

or

```
$ export SSL_CERT_DIR=/path/to/server/dir
```

Installing the Certificate on Windows

1. Right-click on the certificate file and click Install Certificate.

2. Choose to install the certificate as the Current User or Local Machine. Choose the Trusted Root Certification Authorities as the certification store.
Using cf CLI Plugins

The Cloud Foundry Command Line Interface (cf CLI) includes plugin functionality. These plugins enable developers to add custom commands to the cf CLI. You can install and use plugins that Cloud Foundry developers and third-party developers create. You can review the Cloud Foundry Community CLI Plugin page for a current list of community-supported plugins. You can find information about submitting your own plugin to the community in the Cloud Foundry Community CLI plugin repository on GitHub.

警告: Plugins are not vetted in any way, including for security or functionality. Use plugins at your own risk.

The cf CLI identifies a plugin by its binary filename, its developer-defined plugin name, and the commands that the plugin provides. You use the binary filename only to install a plugin. You use the plugin name or a command for any other action.

Note: The cf CLI uses case-sensitive commands, but plugin management commands accept plugin and repository names irrespective of their casing.

Changing the Plugin Directory

By default, the cf CLI stores plugins on your workstation in $CF_HOME/.cf/plugins, which defaults to $HOME/.cf/plugins. To change the root directory of this path from $CF_HOME, set the CF_PLUGIN_HOME environment variable. The cf CLI appends .cf/plugins to the CF_PLUGIN_HOME path that you specify and stores plugins in that location.

For example, if you set CF_PLUGIN_HOME to /my-folder, cf CLI stores plugins in /my-folder/.cf/plugins.

Installing a Plugin

1. Download a binary or the source code for a plugin from a trusted provider.

   Note: The cf CLI requires a binary file compiled from source code written in Go. If you download source code, you must compile the code to create a binary.

2. Run cf install-plugin BINARY-Filename to install a plugin. Replace BINARY-Filename with the path to and name of your binary file.

   Note: You cannot install a plugin that has the same name or that uses the same command as an existing plugin. You will be prompted to uninstall the existing plugin.

   Note: The cf CLI prohibits you from implementing any plugin that uses a native cf CLI command name or alias. For example, if you attempt to install a third-party plugin that includes the command cf push, the cf CLI halts the installation.

Running a Plugin Command

Use the contents of the cf CLI plugin management and Commands offered by installed plugins sections to manage plugins and run plugin commands.

1. Run cf plugins to list all installed plugins and all commands that the plugins provide.

2. Run cf PLUGIN-COMMAND to execute a plugin command.

Checking for Plugin Updates

Run cf plugins -- outdated to check all registered plugin repositories for newer versions of currently installed plugins.
Example:

```
$ cf plugins --outdated
Searching CF-Community, company-repo for newer versions of installed plugins...
plugin   version latest version
coffeemaker 1.1.2   1.2.0
Use `cf install-plugin` to update a plugin to the latest version.
```

For more information about the `cf plugins` command, see [cf plugins](https://docs.cloudfoundry.org/).

### Uninstalling a Plugin

Use the `PLUGIN-NAME` to remove a plugin, not the `BINARY-FILENAME`.

1. Run `cf plugins` to view the names of all installed plugins.
2. Run `cf uninstall-plugin PLUGIN-NAME` to remove a plugin.

### Adding a Plugin Repository

Run `cf add-plugin-repo REPOSITORY-NAME-URL` to add a plugin repository.

Example:

```
$ cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
https://plugins.cloudfoundry.org added as CF-Community
```

### Listing Available Plugin Repositories

Run `cf list-plugin-repos` to view your available plugin repositories.

Example:

```
$ cf list-plugin-repos
OK
Repo Name       Url
    CF-Community https://plugins.cloudfoundry.org
```

### Listing All Plugins by Repository

Run `cf repo-plugins` to show all plugins from all available repositories.

### Troubleshooting

The `cf` CLI provides the following error messages to help you troubleshoot installation and usage issues. Third-party plugins can provide their own error messages.

#### Permission Denied

If you receive a `permission denied` error message, you lack required permissions to the plugin. You must have `read` and `execute` permissions to the plugin binary file.
Plugin Command Collision

Plugin names and commands must be unique. The CLI displays an error message if you attempt to install a plugin with a non-unique name or command.

If the plugin has the same name or command as a currently installed plugin, you must first uninstall the existing plugin to install the new plugin.

If the plugin has a command with the same name as a native cf CLI command or alias, you cannot install the plugin.
Developing cf CLI Plugins

Users can create and install Cloud Foundry Command Line Interface (cf CLI) plugins to provide custom commands. These plugins can be submitted and shared to the CF Community repository.

Requirements

Using plugins requires cf CLI v.6.7 or higher. Refer to the Installing the Cloud Foundry Command Line Interface topic for information about downloading, installing, and uninstalling the cf CLI.

Installing the Architecture

1. Implement the predefined plugin interface.
2. Clone the template repository. You will need the basic GO plugin.

Initializing the Plugin

To initialize a plugin, call plugin.Start(new(MyPluginStruct)) from within the main() method of your plugin. The plugin.Start(...) function requires a new reference to the struct that implements the defined interface.

Invoking cf CLI Commands

Invoke cf CLI commands with cliConnection.CliCommand([]args) from within a plugin’s Run(...) method. The Run(...) method receives the cliConnection as its first argument. The cliConnection.CliCommand([]args) returns the output printed by the command and an error.

The output is returned as a slice of strings. The error will be present if the call to the cf CLI command fails.

For more information, see the API commands documentation.

Installing a Plugin

To install a plugin, run cf install-plugin PATH_TO_PLUGIN_BINARY.

For additional information about developing plugins, see the plugin development guide.
Cloud Foundry CLI Reference Guide

Name
cf - A command line tool to interact with Cloud Foundry

Usage
cf [global options] command [arguments…] [command options]

Version
6.43.0+815ea2f3d.2019-02-20

Getting Started

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>Show help</td>
</tr>
<tr>
<td>version</td>
<td>Print the version</td>
</tr>
<tr>
<td>login</td>
<td>Log user in</td>
</tr>
<tr>
<td>logout</td>
<td>Log user out</td>
</tr>
<tr>
<td>passwd</td>
<td>Change user password</td>
</tr>
<tr>
<td>target</td>
<td>Set or view the targeted org or space</td>
</tr>
<tr>
<td>api</td>
<td>Set or view target api url</td>
</tr>
<tr>
<td>auth</td>
<td>Authenticate non-interactively</td>
</tr>
</tbody>
</table>

Apps

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apps</td>
<td>List all apps in the target space</td>
</tr>
<tr>
<td>app</td>
<td>Display health and status for an app</td>
</tr>
<tr>
<td>push</td>
<td>Push a new app or sync changes to an existing app</td>
</tr>
<tr>
<td>scale</td>
<td>Change or view the instance count, disk space limit, and memory limit for an app</td>
</tr>
<tr>
<td>delete</td>
<td>Delete an app</td>
</tr>
<tr>
<td>rename</td>
<td>Rename an app</td>
</tr>
<tr>
<td>start</td>
<td>Start an app</td>
</tr>
<tr>
<td>stop</td>
<td>Stop an app</td>
</tr>
<tr>
<td>restart</td>
<td>Stop all instances of the app, then start them again. This causes downtime.</td>
</tr>
<tr>
<td>restage</td>
<td>Recreate the app’s executable artifact using the latest pushed app files and the latest environment (variables, service bindings, buildpack, stack, etc.). This action will cause app downtime.</td>
</tr>
<tr>
<td>restart-app-instance</td>
<td>Terminate, then restart an app instance</td>
</tr>
<tr>
<td>run-task</td>
<td>Run a one-off task on an app</td>
</tr>
<tr>
<td>tasks</td>
<td>List tasks of an app</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>terminate-task</td>
<td>Terminate a running task of an app</td>
</tr>
<tr>
<td>events</td>
<td>Show recent app events</td>
</tr>
<tr>
<td>files</td>
<td>Print out a list of files in a directory or the contents of a specific file of an app running on the DEA backend</td>
</tr>
<tr>
<td>logs</td>
<td>Tail or show recent logs for an app</td>
</tr>
<tr>
<td>env</td>
<td>Show all env variables for an app</td>
</tr>
<tr>
<td>set-env</td>
<td>Set an env variable for an app</td>
</tr>
<tr>
<td>unset-env</td>
<td>Remove an env variable from an app</td>
</tr>
<tr>
<td>stacks</td>
<td>List all stacks (a stack is a pre-built file system, including an operating system, that can run apps)</td>
</tr>
<tr>
<td>stack</td>
<td>Show information for a stack (a stack is a pre-built file system, including an operating system, that can run apps)</td>
</tr>
<tr>
<td>copy-source</td>
<td>Copies the source code of an application to another existing application (and restarts that application)</td>
</tr>
<tr>
<td>create-app-manifest</td>
<td>Create an app manifest for an app that has been pushed successfully</td>
</tr>
<tr>
<td>get-health-check</td>
<td>Show the type of health check performed on an app</td>
</tr>
<tr>
<td>set-health-check</td>
<td>Change type of health check performed on an app</td>
</tr>
<tr>
<td>enable-ssh</td>
<td>Enable ssh for the application</td>
</tr>
<tr>
<td>disable-ssh</td>
<td>Disable ssh for the application</td>
</tr>
<tr>
<td>ssh-enabled</td>
<td>Reports whether SSH is enabled on an application container instance</td>
</tr>
<tr>
<td>ssh</td>
<td>SSH to an application container instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>marketplace</td>
<td>List available offerings in the marketplace</td>
</tr>
<tr>
<td>services</td>
<td>List all service instances in the target space</td>
</tr>
<tr>
<td>service</td>
<td>Show service instance info</td>
</tr>
<tr>
<td>create-service</td>
<td>Create a service instance</td>
</tr>
<tr>
<td>update-service</td>
<td>Update a service instance</td>
</tr>
<tr>
<td>delete-service</td>
<td>Delete a service instance</td>
</tr>
<tr>
<td>rename-service</td>
<td>Rename a service instance</td>
</tr>
<tr>
<td>create-service-key</td>
<td>Create key for a service instance</td>
</tr>
<tr>
<td>service-keys</td>
<td>List keys for a service instance</td>
</tr>
<tr>
<td>service-key</td>
<td>Show service key info</td>
</tr>
<tr>
<td>delete-service-key</td>
<td>Delete a service key</td>
</tr>
<tr>
<td>bind-service</td>
<td>Bind a service instance to an app</td>
</tr>
<tr>
<td>unbind-service</td>
<td>Unbind a service instance from an app</td>
</tr>
<tr>
<td>bind-route-service</td>
<td>Bind a service instance to an HTTP route</td>
</tr>
<tr>
<td>unbind-route-service</td>
<td>Unbind a service instance from an HTTP route</td>
</tr>
<tr>
<td>create-user-provided-service</td>
<td>Make a user-provided service instance available to CF apps</td>
</tr>
<tr>
<td>update-user-provided-service</td>
<td>Update user-provided service instance</td>
</tr>
<tr>
<td>share-service</td>
<td>Share a service instance with another space</td>
</tr>
<tr>
<td>unshare-service</td>
<td>Unshare a shared service instance from a space</td>
</tr>
</tbody>
</table>
## Orgs

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgs 👈</td>
<td>List all orgs</td>
</tr>
<tr>
<td>org 👈</td>
<td>Show org info</td>
</tr>
<tr>
<td>create-org 👈</td>
<td>Create an org</td>
</tr>
<tr>
<td>delete-org 👈</td>
<td>Delete an org</td>
</tr>
<tr>
<td>rename-org 👈</td>
<td>Rename an org</td>
</tr>
</tbody>
</table>

## Spaces

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spaces 👈</td>
<td>List all spaces in an org</td>
</tr>
<tr>
<td>space 👈</td>
<td>Show space info</td>
</tr>
<tr>
<td>create-space 👈</td>
<td>Create a space</td>
</tr>
<tr>
<td>delete-space 👈</td>
<td>Delete a space</td>
</tr>
<tr>
<td>rename-space 👈</td>
<td>Rename a space</td>
</tr>
<tr>
<td>allow-space-ssh 👈</td>
<td>Allow SSH access for the space</td>
</tr>
<tr>
<td>disallow-space-ssh 👈</td>
<td>Disallow SSH access for the space</td>
</tr>
<tr>
<td>space-ssh-allowed 👈</td>
<td>Reports whether SSH is allowed in a space</td>
</tr>
</tbody>
</table>

## Domains

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domains 👈</td>
<td>List domains in the target org</td>
</tr>
<tr>
<td>create-domain 👈</td>
<td>Create a domain in an org for later use</td>
</tr>
<tr>
<td>delete-domain 👈</td>
<td>Delete a domain</td>
</tr>
<tr>
<td>create-shared-domain 👈</td>
<td>Create a domain that can be used by all orgs (admin-only)</td>
</tr>
<tr>
<td>delete-shared-domain 👈</td>
<td>Delete a shared domain</td>
</tr>
<tr>
<td>router-groups 👈</td>
<td>List router groups</td>
</tr>
</tbody>
</table>

## Routes

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>routes 👈</td>
<td>List all routes in the current space or the current organization</td>
</tr>
<tr>
<td>create-route 👈</td>
<td>Create a url route in a space for later use</td>
</tr>
<tr>
<td>check-route 👈</td>
<td>Perform a simple check to determine whether a route currently exists or not</td>
</tr>
<tr>
<td>map-route 👈</td>
<td>Add a url route to an app</td>
</tr>
<tr>
<td>unmap-route 👈</td>
<td>Remove a url route from an app</td>
</tr>
<tr>
<td>delete-route 👈</td>
<td>Delete a route</td>
</tr>
<tr>
<td>delete-orphaned-routes 👈</td>
<td>Delete all orphaned routes (i.e. those that are not mapped to an app)</td>
</tr>
</tbody>
</table>
### Network Policies

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network-policies</td>
<td>List direct network traffic policies</td>
</tr>
<tr>
<td>add-network-policy</td>
<td>Create policy to allow direct network traffic from one app to another</td>
</tr>
<tr>
<td>remove-network-policy</td>
<td>Remove network traffic policy of an app</td>
</tr>
</tbody>
</table>

### Buildpacks

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildpacks</td>
<td>List all buildpacks</td>
</tr>
<tr>
<td>create-buildpack</td>
<td>Create a buildpack</td>
</tr>
<tr>
<td>update-buildpack</td>
<td>Update a buildpack</td>
</tr>
<tr>
<td>rename-buildpack</td>
<td>Rename a buildpack</td>
</tr>
<tr>
<td>delete-buildpack</td>
<td>Delete a buildpack</td>
</tr>
</tbody>
</table>

### User Admin

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create-user</td>
<td>Create a new user</td>
</tr>
<tr>
<td>delete-user</td>
<td>Delete a user</td>
</tr>
<tr>
<td>org-users</td>
<td>Show org users by role</td>
</tr>
<tr>
<td>set-org-role</td>
<td>Assign an org role to a user</td>
</tr>
<tr>
<td>unset-org-role</td>
<td>Remove an org role from a user</td>
</tr>
<tr>
<td>space-users</td>
<td>Show space users by role</td>
</tr>
<tr>
<td>set-space-role</td>
<td>Assign a space role to a user</td>
</tr>
<tr>
<td>unset-space-role</td>
<td>Remove a space role from a user</td>
</tr>
</tbody>
</table>

### Org Admin

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quotas</td>
<td>List available usage quotas</td>
</tr>
<tr>
<td>quota</td>
<td>Show quota info</td>
</tr>
<tr>
<td>set-quota</td>
<td>Assign a quota to an org</td>
</tr>
<tr>
<td>create-quota</td>
<td>Define a new resource quota</td>
</tr>
<tr>
<td>delete-quota</td>
<td>Delete a quota</td>
</tr>
<tr>
<td>update-quota</td>
<td>Update an existing resource quota</td>
</tr>
<tr>
<td>share-private-domain</td>
<td>Share a private domain with an org</td>
</tr>
<tr>
<td>unshare-private-domain</td>
<td>Unshare a private domain with an org</td>
</tr>
</tbody>
</table>

### Space Admin

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>space-quotas</td>
<td>List available space resource quotas</td>
</tr>
<tr>
<td>space-quota</td>
<td>Show space quota info</td>
</tr>
<tr>
<td>create-space-quota</td>
<td>Define a new space resource quota</td>
</tr>
<tr>
<td>update-space-quota</td>
<td>Update an existing space quota</td>
</tr>
<tr>
<td>delete-space-quota</td>
<td>Delete a space quota definition and unassign the space quota from all spaces</td>
</tr>
<tr>
<td>set-space-quota</td>
<td>Assign a space quota definition to a space</td>
</tr>
<tr>
<td>unset-space-quota</td>
<td>Unassign a quota from a space</td>
</tr>
</tbody>
</table>

**Service Admin**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service-auth-tokens</td>
<td>List service auth tokens</td>
</tr>
<tr>
<td>create-service-auth-token</td>
<td>Create a service auth token</td>
</tr>
<tr>
<td>update-service-auth-token</td>
<td>Update a service auth token</td>
</tr>
<tr>
<td>delete-service-auth-token</td>
<td>Delete a service auth token</td>
</tr>
<tr>
<td>service-brokers</td>
<td>List service brokers</td>
</tr>
<tr>
<td>create-service-broker</td>
<td>Create a service broker</td>
</tr>
<tr>
<td>update-service-broker</td>
<td>Update a service broker</td>
</tr>
<tr>
<td>delete-service-broker</td>
<td>Delete a service broker</td>
</tr>
<tr>
<td>rename-service-broker</td>
<td>Rename a service broker</td>
</tr>
<tr>
<td>migrate-service-instances</td>
<td>Migrate service instances from one service plan to another</td>
</tr>
<tr>
<td>purge-service-offering</td>
<td>Recursively remove a service and child objects from Cloud Foundry database</td>
</tr>
<tr>
<td>purge-service-instance</td>
<td>Recursively remove a service instance and child objects from Cloud Foundry</td>
</tr>
<tr>
<td>service-access</td>
<td>List service access settings</td>
</tr>
<tr>
<td>enable-service-access</td>
<td>Enable access to a service or service plan for one or all orgs</td>
</tr>
<tr>
<td>disable-service-access</td>
<td>Disable access to a service or service plan for one or all orgs</td>
</tr>
</tbody>
</table>

**Security Group**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>security-group</td>
<td>Show a single security group</td>
</tr>
<tr>
<td>security-groups</td>
<td>List all security groups</td>
</tr>
<tr>
<td>create-security-group</td>
<td>Create a security group</td>
</tr>
<tr>
<td>update-security-group</td>
<td>Update a security group</td>
</tr>
<tr>
<td>delete-security-group</td>
<td>Deletes a security group</td>
</tr>
<tr>
<td>bind-security-group</td>
<td>Bind a security group to a particular space, or all existing spaces of an org</td>
</tr>
<tr>
<td>unbind-security-group</td>
<td>Unbind a security group from a space</td>
</tr>
<tr>
<td>bind-staging-security-group</td>
<td>Bind a security group to the list of security groups to be used for staging applications</td>
</tr>
<tr>
<td>staging-security-groups</td>
<td>List security groups in the staging set for applications</td>
</tr>
<tr>
<td>unbind-staging-security-group</td>
<td>Unbind a security group from the set of security groups for staging applications</td>
</tr>
</tbody>
</table>
### Environment Variable Groups

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>running-environment-variable-group</td>
<td>Retrieve the contents of the running environment variable group</td>
</tr>
<tr>
<td>staging-environment-variable-group</td>
<td>Retrieve the contents of the staging environment variable group</td>
</tr>
<tr>
<td>set-staging-environment-variable-group</td>
<td>Pass parameters as JSON to create a staging environment variable group</td>
</tr>
<tr>
<td>set-running-environment-variable-group</td>
<td>Pass parameters as JSON to create a running environment variable group</td>
</tr>
</tbody>
</table>

### Isolation Segments

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isolation-segments</td>
<td>List all isolation segments</td>
</tr>
<tr>
<td>create-isolation-segment</td>
<td>Create an isolation segment</td>
</tr>
<tr>
<td>delete-isolation-segment</td>
<td>Delete an isolation segment</td>
</tr>
<tr>
<td>enable-org-isolation</td>
<td>Entitle an organization to an isolation segment</td>
</tr>
<tr>
<td>disable-org-isolation</td>
<td>Revoke an organization’s entitlement to an isolation segment</td>
</tr>
<tr>
<td>set-org-default-isolation-segment</td>
<td>Set the default isolation segment used for apps in spaces in an org</td>
</tr>
<tr>
<td>reset-org-default-isolation-segment</td>
<td>Reset the default isolation segment used for apps in spaces of an org</td>
</tr>
<tr>
<td>set-space-isolation-segment</td>
<td>Assign the isolation segment for a space</td>
</tr>
<tr>
<td>reset-space-isolation-segment</td>
<td>Reset the space’s isolation segment to the org default</td>
</tr>
</tbody>
</table>

### Feature Flags

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature-flags</td>
<td>Retrieve list of feature flags with status</td>
</tr>
<tr>
<td>feature-flag</td>
<td>Retrieve an individual feature flag with status</td>
</tr>
<tr>
<td>enable-feature-flag</td>
<td>Allow use of a feature</td>
</tr>
<tr>
<td>disable-feature-flag</td>
<td>Prevent use of a feature</td>
</tr>
</tbody>
</table>

### Advanced

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>curl</td>
<td>Executes a request to the targeted API endpoint</td>
</tr>
<tr>
<td>config</td>
<td>Write default values to the config</td>
</tr>
<tr>
<td>oauth-token</td>
<td>Retrieve and display the OAuth token for the current session</td>
</tr>
<tr>
<td>ssh-code</td>
<td>Get a one time password for ssh clients</td>
</tr>
</tbody>
</table>

### Add/remove Plugin Repository

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>remove-plugin-repo</td>
<td>Remove a plugin repository</td>
</tr>
<tr>
<td>list-plugin-repos</td>
<td>List all the added plugin repositories</td>
</tr>
<tr>
<td>repo-plugins</td>
<td>List all available plugins in specified repository or in all added repositories</td>
</tr>
</tbody>
</table>

Add/remove Plugin

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plugins</td>
<td>List commands of installed plugins</td>
</tr>
<tr>
<td>install-plugin</td>
<td>Install CLI plugin</td>
</tr>
<tr>
<td>uninstall-plugin</td>
<td>Uninstall CLI plugin</td>
</tr>
</tbody>
</table>

Environment Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF_COLOR=false</td>
<td>Do not colorize output</td>
</tr>
<tr>
<td>CF_DIAL_TIMEOUT=5</td>
<td>Max wait time to establish a connection, including name resolution, in seconds</td>
</tr>
<tr>
<td>CF_HOME=path/to/dir/</td>
<td>Override path to default config directory</td>
</tr>
<tr>
<td>CF_PLUGIN_HOME=path/to/dir/</td>
<td>Override path to default plugin config directory</td>
</tr>
<tr>
<td>CF_TRACE=true</td>
<td>Print API request diagnostics to stdout</td>
</tr>
<tr>
<td>CF_TRACE=path/to/trace.log</td>
<td>Append API request diagnostics to a log file</td>
</tr>
<tr>
<td>all_proxy=proxy.example.com:8080</td>
<td>Specify a proxy server to enable proxying for all requests</td>
</tr>
<tr>
<td>https_proxy=proxy.example.com:8080</td>
<td>Enable proxying for HTTP requests</td>
</tr>
</tbody>
</table>

Global Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help, -h</td>
<td>Show help</td>
</tr>
<tr>
<td>-v</td>
<td>Print API request diagnostics to stdout</td>
</tr>
</tbody>
</table>

Apps (experimental)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3-apps</td>
<td>List all apps in the target space</td>
</tr>
<tr>
<td>v3-create-app</td>
<td>Create a V3 App</td>
</tr>
<tr>
<td>v3-push</td>
<td>Push a new app or sync changes to an existing app</td>
</tr>
<tr>
<td>v3-scale</td>
<td>Change or view the instance count, disk space limit, and memory limit for an app</td>
</tr>
<tr>
<td>v3-delete</td>
<td>Delete a V3 App</td>
</tr>
<tr>
<td>v3-start</td>
<td>Start an app</td>
</tr>
<tr>
<td>v3-stop</td>
<td>Stop an app</td>
</tr>
<tr>
<td>v3-restart</td>
<td>Stop all instances of the app, then start them again. This causes downtime.</td>
</tr>
<tr>
<td>v3-stage</td>
<td>Create a new droplet for an app</td>
</tr>
<tr>
<td>v3-restart-app-instance</td>
<td>Terminate, then instantiate an app instance</td>
</tr>
<tr>
<td>v3-apply-manifest</td>
<td>Applies manifest properties to an application</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>v3-set-droplet</td>
<td>Set the droplet used to run an app</td>
</tr>
<tr>
<td>v3-env</td>
<td>Show all env variables for an app</td>
</tr>
<tr>
<td>v3-set-env</td>
<td>Set an env variable for an app</td>
</tr>
<tr>
<td>v3-unset-env</td>
<td>Remove an env variable from an app</td>
</tr>
<tr>
<td>v3-get-health-check</td>
<td>Show the type of health check performed on an app</td>
</tr>
<tr>
<td>v3-set-health-check</td>
<td>Change type of health check performed on an app's process</td>
</tr>
<tr>
<td>v3-packages</td>
<td>List packages of an app</td>
</tr>
<tr>
<td>v3-create-package</td>
<td>Uploads a V3 Package</td>
</tr>
<tr>
<td>v3-ssh</td>
<td>SSH to an application container instance</td>
</tr>
</tbody>
</table>
Using Experimental cf CLI Commands

Page last updated:

This topic describes how to use the experimental Cloud Foundry Command Line Interface (cf CLI) commands offered by the Cloud Controller V3 API.

These commands provide developers with the ability to better orchestrate app deployment workflows. New features include the deployment and management of apps with multiple processes and uploading and staging multiple versions of a single app.

The experimental commands described in this topic require Pivotal Cloud Foundry (PCF) 1.12+ and the cf CLI v6.30.0+.

For more information about cf CLI commands, see the Cloud Foundry CLI Reference Guide. For more information about the Cloud Controller V3 API, see the API documentation.

Note: Because these commands are experimental, they are not guaranteed to be available or compatible in subsequent cf CLI releases.

Overview

The new commands include a `v3-` prefix. While the syntax of some experimental commands is based on the existing cf CLI, these commands call the V3 API and support new flags to unlock additional features. Other commands expose the new primitives of apps, such as by performing operations on an app’s packages and droplets.

In the V2 APIs, running and staging an app are tightly coupled operations. As a result, an app cannot be staging and running at the same time. The V3 APIs offer developers more granular control over the uploading, staging, and running of an app.

Commands

Consult the following table for a description of the experimental commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3-apply-manifest</td>
<td>Applies manifest properties to an app</td>
</tr>
<tr>
<td>v3-apps</td>
<td>Lists all apps in the target space</td>
</tr>
<tr>
<td>v3-create-app</td>
<td>Creates a V3 app</td>
</tr>
<tr>
<td>v3-create-package</td>
<td>Uploads a V3 package</td>
</tr>
<tr>
<td>v3-delete</td>
<td>Deletes a V3 app</td>
</tr>
<tr>
<td>v3-droplets</td>
<td>Lists droplets of an app</td>
</tr>
<tr>
<td>v3-env</td>
<td>Shows all environment variables for an app</td>
</tr>
<tr>
<td>v3-get-health-check</td>
<td>Shows the type of health check performed on an app</td>
</tr>
<tr>
<td>v3-packages</td>
<td>Lists packages of an app</td>
</tr>
<tr>
<td>v3-push</td>
<td>Pushes a new app or syncs changes to an existing app</td>
</tr>
<tr>
<td>v3-restart</td>
<td>Stops all instances of an app and then starts them again, which may cause downtime</td>
</tr>
<tr>
<td>v3-restart-app-instance</td>
<td>Terminates and then instantiates an app instance</td>
</tr>
<tr>
<td>v3-set-droplet</td>
<td>Sets the droplet used to run an app</td>
</tr>
<tr>
<td>v3-set-env</td>
<td>Sets an environment variable for an app</td>
</tr>
<tr>
<td>v3-set-health-check</td>
<td>Changes type of health check performed on an app’s process</td>
</tr>
<tr>
<td>v3-ssh</td>
<td>SSH to an app container instance</td>
</tr>
<tr>
<td>v3-stage</td>
<td>Creates a new droplet for an app</td>
</tr>
<tr>
<td>v3-start</td>
<td>Starts an app</td>
</tr>
<tr>
<td>v3-stop</td>
<td>Stops an app</td>
</tr>
<tr>
<td>v3-unset-env</td>
<td>Removes an environment variable from an app</td>
</tr>
</tbody>
</table>
Considerations

Keep in mind the following considerations when using the experimental commands:

- `v3-push` supports only a subset of features of `push`. In particular, it does not support app manifests.
- For some commands, such as `set-env`, `ssh`, and `bind-service`, no new V3 version exists. In those cases, use the old commands.
- You can use V3 and old commands together, but some combinations may give unexpected results. For example, if you use V3 commands to create an app with a package but it is not staged, or you use `v3-push` to push an app but it fails to stage, the old `apps` command does not return the app.

Declaring Multiple App Processes with a Procfile

Developers can supply a Procfile with their app in order to run multiple long-lived processes using a single codebase.

To use a Procfile, include it in the root of your app directory and push your app.

For more information about Procfiles, see the About Procfiles section of the Production Server Configuration topic.
Developer Guide

This guide provides instructions for deploying and managing apps and services.

Check out the 15-minute Getting Started with PCF tutorial for learning Pivotal Cloud Foundry app deployment concepts.

See the following topics:

- cf push
- SSH for Apps and Services
- Routes and Domains
- Managing Services
- Streaming App Logs
- Managing Apps with the cf CLI
- Cloud Foundry Environment Variables
- Cloud Controller API Client Libraries
- Considerations for Designing and Running and App in the Cloud
**cf push**

This topic provides an overview of the documentation available for `cf push`.

### How to Use cf push

The following topics provide procedures for deploying apps with `cf push`:

- Deploying an App
- Deploying with App Manifests
  - App Manifest Attribute Reference
- Deploying an App with Docker
- Deploying a Large App
- Starting, Restarting, and Restaging Apps
- Pushing an Application with Multiple Buildpacks
- Pushing an App with Multiple Processes (Beta)
- Running `cf push` Sub-Step Commands (Beta)
- Rolling App Deployments (Beta)

### How to Troubleshoot

For information about troubleshooting when running `cf push`, see Troubleshooting App Deployment and Health.

### How cf push Works

The following topics provide information about how `cf push` works:

- The Push section of Getting Started with the cf CLI.
- App Container Lifecycle
- How Apps Are Staged
Deploying an App

Page last updated:

Note: See the [buildpacks](#) documentation for deployment guides specific to your app language or framework, such as the [Getting Started Deploying Ruby on Rails Apps](#) guide.

---

Overview of Deployment Process

You deploy an app to Cloud Foundry by running a `cf push` command from the Cloud Foundry Command Line Interface (cf CLI). Refer to the [Installing the cf CLI](#) topic for more information. Between the time that you run `cf push` and the time that the app is available, Cloud Foundry performs the following tasks:

- Uploads and stores app files
- Examines and stores app metadata
- Creates a “droplet” (the Cloud Foundry unit of execution) for the app
- Selects an appropriate Diego cell to run the droplet
- Starts the app

For more information about the lifecycle of an app, see the [App Container Lifecycle](#) topic.

An app that uses services, such as a database, messaging, or email server, is not fully functional until you provision the service and, if required, bind the service to the app. For more information about services, see the [Services Overview](#) topic.

---

Step 1: Prepare to Deploy

Before you deploy your app to Cloud Foundry, make sure that:

- Your app is cloud-ready. Cloud Foundry behaviors related to file storage, HTTP sessions, and port usage may require modifications to your app.
- All required app resources are uploaded. For example, you may need to include a database driver.
- Extraneous files and artifacts are excluded from upload. You should explicitly exclude extraneous files that reside within your app directory structure, particularly if your app is large.
- An instance of every service that your app needs has been created.
- Your Cloud Foundry instance supports the type of app you are going to deploy, or you have the URL of an externally available buildpack that can stage the app.

For help preparing to deploy your app, see:

- [Considerations for Designing and Running an App in the Cloud](#)
- [Buildpacks](#)

---

Step 2: Know Your Credentials and Target

Before you can push your app to Cloud Foundry you need to know:

- The API endpoint for your Cloud Foundry instance. Also known as the target URL, this is the [URL of the Cloud Controller in your PAS instance](#).
- Your username and password for your Cloud Foundry instance.
- The organization and space where you want to deploy your app. A Cloud Foundry workspace is organized into organizations, and within them, spaces. As a Cloud Foundry user, you have access to one or more organizations and spaces.

---

Step 3: (Optional) Configure Domains

Cloud Foundry directs requests to an app using a route, which is a URL made up of a host and a domain.
The name of an app is the default host for that app, unless you specify the host name with the `-n` flag.

Every app is deployed to an app space that belongs to a domain. Every Cloud Foundry instance has a default domain defined. You can specify a non-default, or custom, domain when deploying, provided that the domain is registered and is mapped to the organization which contains the target app space.

**Note:** CF allows app names, but not app URLs, to include underscores. CF converts underscores to hyphens when setting a default app URL from an app name.

The URL for your app must be unique from other apps hosted by PAS. Use the following options with the `cf CLI` to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words

**Note:** Use `cf help push` to view other options for this command.

For more information about domains, see Routes and Domains.

### Step 4: Determine Deployment Options

Before you deploy, you need to decide on the following:

- **Name:** You can use any series of alpha-numeric characters as the name of your app.
- **Instances:** Generally speaking, the more instances you run, the less downtime your app will experience. If your app is still in development, running a single instance can simplify troubleshooting. For any production app, we recommend a minimum of two instances.
- **Memory Limit:** The maximum amount of memory that each instance of your app can consume. If an instance exceeds this limit, Cloud Foundry restarts the instance.

**Note:** Initially, Cloud Foundry immediately restarts any instances that exceed the memory limit. If an instance repeatedly exceeds the memory limit in a short period of time, Cloud Foundry delays restarting the instance.

- **Start Command:** This is the command that Cloud Foundry uses to start each instance of your app. This start command varies by app framework.
- **Subdomain (host) and Domain:** The route, which is the combination of subdomain and domain, must be globally unique. This is true whether you specify a portion of the route or allow Cloud Foundry to use defaults.
- **Services:** Apps can bind to services such as databases, messaging, and key-value stores. Apps are deployed into app spaces. An app can only bind to a service that has an existing instance in the target app space.

### Define Deployment Options

You can define deployment options on the command line, in a manifest file, or both together. See Deploying with App Manifests to learn how app settings change from push to push, and how command-line options, manifests, and commands like `cf scale` interact.

When you deploy an app while it is running, Cloud Foundry stops all instances of that app and then deploys. Users who try to run the app get a “404 not found” message while `cf push` runs. Stopping all instances is necessary to prevent two versions of your code from running at the same time. A worst-case example would be deploying an update that involved a database schema migration, because instances running the old code would not work and users could lose data.

Cloud Foundry uploads all app files except version control files and folders with names such as `.svn`, `.git`, and `_darcs`. To exclude other files from upload, specify them in a `.cfignore` file in the directory where you run the push command. For more information, see the Ignore Unnecessary Files When Pushing section of the Considerations for Designing and Running an App in the Cloud topic.

For more information about the manifest file, see the Deploying with App Manifests topic.

### Configure Pre-Runtime Hooks

**Note:** The Java buildpack does not support pre-runtime hooks.
To configure pre-runtime hooks, create a file named `.profile` and place it in the root of your app directory. If the directory includes a `.profile` script, then Cloud Foundry executes it immediately before each instance of your app starts. Because the `.profile` script executes after the buildpack, the script has access to the language runtime environment created by the buildpack.

Note: Your app root directory may also include a `.profile.d` directory that contains bash scripts that perform initialization tasks for the buildpack. Developers should not edit these scripts unless they are using a custom buildpack.

You can use the `.profile` script to perform app-specific initialization tasks, such as setting custom environment variables. Environment variables are key-value pairs defined at the operating system level. These key-value pairs provide a way to configure the apps running on a system. For example, any app can access the LANG environment variable to determine which language to use for error messages and instructions, collating sequences, and date formats.

To set an environment variable, add the appropriate bash commands to your `.profile` file. See the example below.

```bash
# Set the default LANG for your apps
export LANG=en_US.UTF-8
```

Note: If you are using a PHP buildpack version prior to v4.3.18, the buildpack does not execute your PHP app’s `.profile` script. Your PHP app will host the `.profile` script’s contents. This means that any PHP app staged using the affected PHP buildpack versions can leak credentials placed in the `.profile` script.

**Step 5: Push the App**

Run the following command to deploy an app without a manifest:

```bash
cf push APP-NAME
```

If you provide the app name in a manifest, you can reduce the command to `cf push`. See [Deploying with App Manifests](#). Because all you have provided is the name of your app, `cf push` sets the number of instances, amount of memory, and other attributes of your app to the default values. You can also use command-line options to specify these and additional attributes.

The following transcript illustrates how Cloud Foundry assigns default values to app when given a `cf push` command.

Note: When deploying your own apps, avoid generic names like `my-app`. Cloud Foundry uses the app name to compose the route to the app, and deployment fails unless the app has a globally unique route.
Step 6: (Optional) Configure Service Connections

If you bound a service to the app that you deployed, you might need to configure your app with the service URL and credentials. For more information, see the specific documentation for your app framework:

- Ruby
- Node.js
- Spring
- Grails

Step 7: Troubleshoot Deployment Problems

If your app does not start on Cloud Foundry, first ensure that your app can run locally.

You can troubleshoot your app in the cloud using the cf CLI. See Troubleshoot App Deployment and Health.
Deploying with App Manifests

Page last updated:

This topic provides basic procedures and guidance for deploying apps with a manifest file.

Overview

Manifests provide consistency and reproducibility, and can help you automate deploying apps. Both manifests and command line options allow you to override the default attribute values of `cf push`. These attributes include things like number of instances, disk space limit, and memory limit.

`cf push` follows rules of precedence when setting attribute values:

- Manifests override most recent values, including defaults and values set by commands such as `cf scale`.
- Command line options override manifests.

For a full list of attributes you can specify in an app manifest, see App Manifest Attribute Reference.

Deploy an App with a Manifest

The following procedure describes how to deploy an app with a minimal manifest. For more information about manifest format and attributes, see App Manifest Attribute Reference.

To deploy an app with a manifest, do the following:

1. Create a `manifest.yml` file in the root directory of your app.

   ![Note](https://via.placeholder.com/150)
   **Note:** By default, the `cf push` command looks for a manifest file in the app directory. However, you can create a manifest file in a different location and use the `-f` flag when you run `cf push`.

2. Add the following content to the file, replacing `MY-APP` with the name of your app:

   ```yaml
   ---
   applications:
     - name: MY-APP
   ```

3. Run `cf push`. If you specify any command line options, they override the values specified in the manifest. See the following section.

Deploy Multiple Apps with One Manifest

This procedure describes how to deploy multiple apps with a minimal manifest. For more information about manifest format and attributes, see App Manifest Attribute Reference.

General Rules

Follow these general rules when deploying multiple apps with one manifest:

- Use a `no-route` line in the description of any app that provides background services to another app.
- Do not use any command line options with `cf push`, with the following exception:
  - If your manifest is not named `manifest.yml` or not in the current working directory, use the `-f` command line option.
- To push a single app rather than all of the apps described in the manifest, provide the desired app name by running `cf push my-app`.

Procedure
To deploy multiple apps with a manifest, do the following:

1. Create a `manifest.yml` file in the directory that contains the apps.

2. Add each app and its directory to the file. CF pushes the apps in the order specified in the manifest.

3. Run `cf push` from the directory that contains the apps and the manifest.

---

```
applications:
  - name: APP-ONE
    path: ./APP-ONE-DIRECTORY
  - name: APP-TWO
    path: ./APP-TWO-DIRECTORY
```

Where:

- **APP-ONE** is the name of the first app you want CF to push
- **APP-ONE-DIRECTORY** is the directory containing the first app
- **APP-TWO** is the name of the second app you want CF to push
- **APP-TWO-DIRECTORY** is the directory containing the second app

Note: Each app must be in a subdirectory under the same parent directory.

Note: If you push multiple apps using a manifest and one fails to deploy, CF does not attempt to push apps specified after the app that failed.
App Manifest Attribute Reference

This topic describes manifest formatting and provides a full list of attributes available for app manifests. You can use it alongside the Deploying with App Manifests topic, which provides basic procedures and guidance for deploying apps with manifests.

Manifest Format

Manifests are written in YAML. The manifest below illustrates some YAML conventions, as follows:

- The manifest begins with three dashes.
- The `applications` block begins with a heading followed by a colon.
- The app `name` is preceded by a single dash and one space.
- Subsequent lines in the block are indented two spaces to align with `name`.

```
---
applications:
  - name: my-app
    memory: 512M
    instances: 2
```

Note: If your app name begins with the dash character (`-`), you cannot interact with the app using the cf CLI. This is because the cf CLI interprets the dash as a flag.

Add Variables to a Manifest

You can use variables to create app manifests with values shared across all applicable environments in combination with references to environment-specific differences defined in separate files.

In addition, using variables enables you to store sensitive data in a separate file that the app manifest would reference, making the security sensitive data easier to manage and keep secure.

To add variables to an app manifest, do the following:

1. Create a file called `vars.yml`.
2. Add attributes to your `vars.yml` file. See the following example:

   ```
   instances: 2
   memory: 1G
   ```
3. Add the variables to your app manifest file using the following format: `((VARIABLE-NAME))`. See the following example:

   ```
   ---
   applications:
     - name: test-app
       instances: ((instances))
       memory: ((memory))
       buildpack: go_buildpack
       env:
         GOPACKAGENAME: go_calls_ruby
         command: go_calls_ruby
   ```

   Note: You can also use variables for partial values. For example, you can specify `host` in your variables file and `route: ((host)).env.com` in your manifest file.

4. Run `cf push`.

   ```
   cf push --vars-file /PATH/vars.yml
   ```

   Where `PATH` is the path to the file you created.
Minimize Duplication with YAML Anchors

**Note:** Promoted content has been deprecated in favor of YAML anchors. For more information, see [Deprecated App Manifest Features](#).

In manifests where multiple apps share settings or services, you may see duplicated content. While the manifests still work, duplication increases the risk of typographical errors, which cause deployments to fail.

You can declare shared configuration using a YAML anchor, which the manifest refers to in app declarations by using an alias.

```yaml
---
defaults: &defaults
  buildpacks:
  - staticfile_buildpack
  memory: 1G

applications:
- name: bigapp
  <<: *defaults
- name: smallapp
  <<: *defaults
  memory: 256M
```

This manifest pushes two apps, smallapp and bigapp, with the staticfile buildpack but with 256M memory for smallapp and 1G for bigapp.

### Application Attributes

This section explains how to describe optional app attributes in manifests. Each of these attributes can also be specified by a command line option. Command line options override the manifest.

**Note:** The route component attributes `domain`, `domains`, `host`, `hosts`, and `no-hostname` have been deprecated in favor of the `routes` attribute. For more information, see [Deprecated App Manifest Features](#).

#### buildpacks

You can refer to a buildpack by name in a manifest or a command line option. The `cf buildpacks` command lists the buildpacks that you can use.

See below for information on referencing buildpacks in a manifest. The command line option that overrides this attribute is `-b`.

- **Custom buildpacks:** If your app requires a custom buildpack, you can use the `buildpack` attribute to specify it in a number of ways:
  - By name: `MY-BUILDPACK`
  - By GitHub URL: `https://github.com/cloudfoundry/java-buildpack.git`
  - By GitHub URL with a branch or tag: `https://github.com/cloudfoundry/java-buildpack.git#v3.3.0` for the `v3.3.0` tag.

```yaml
---
buildpacks:
- buildpack_URL
```

- **Multiple buildpacks:** If you are using multiple buildpacks, you can provide an additional `-b` flag or add an additional value to your manifest:

```yaml
---
...  
buildpacks:
- buildpack_URL
- buildpack_URL
```

**Note:** This feature does not work with [Deprecated App Manifest Features](#).

**Note:** You must specify multiple buildpacks in the correct order: the buildpack will use the app start command given by the final buildpack. See the [multi-buildpack repository](#) for more information.
command

Some languages and frameworks require that you provide a custom command to start an app. Refer to the buildpack documentation to determine if you need to provide a custom start command.

You can provide the custom start command in your app manifest or on the command line. See Starting, Restarting, and Restaging Apps for more information about how Cloud Foundry determines its default start command.

To specify the custom start command in your app manifest, add it in the `command: START-COMMAND` format as the following example shows:

```yaml
---
...command: bundle exec rake VERBOSE=true
```

The start command you specify becomes the default for your app. To return to using the original default start command set by your buildpack, you must explicitly set the attribute to `null` as follows:

```yaml
---
...command: null
```

On the command line, use the `--c` option to specify the custom start command as the following example shows:

```bash
$ cf push my-app --c "bundle exec rake VERBOSE=true"
```

**Note:** The `--c` option with a value of ‘null’ forces `cf push` to use the buildpack start command. See Forcing cf push to use the Buildpack Start Command for more information.

If you override the start command for a Buildpack app, Linux uses `bash -c YOUR-COMMAND` to invoke your app. If you override the start command for a Docker app, Linux uses `sh -c YOUR-COMMAND` to invoke your app. Because of this, if you override a start command, you should prefix `exec` to the final command in your custom composite start command.

An app needs to catch termination signals and clean itself up appropriately. Because of the way that shells manage process trees, the use of custom composite shell commands, particularly those that create child processes using `&`, `&&`, `||`, etc., can prevent your app from receiving signals that are sent to the top level bash process.

To resolve this issue, you can use `exec` to replace the bash process with your own process. For example:

```
bin/rake cf_on_first_instance db:migrate &
bin/rails server-p SPORT -e SRAILS_ENV
```

The process tree is bash -> ruby, so on graceful shutdown only the bash process receives the TERM signal, not the ruby process.

```
bin/rake cf_on_first_instance db:migrate &
exec bin/rails server-p SPORT -e SRAILS_ENV
```

Because of the `exec` prefix included on the final command, the ruby process invoked by rails takes over the bash process managing the execution of the composite command. The process tree is only ruby, so the ruby web server receives the TERM signal and can shutdown gracefully for 10 seconds.

In more complex situations, like making a custom buildpack, you may want to use bash `trap`, `wait`, and backgrounded processes to manage your process tree and shut down apps gracefully. In most situations, however, a well-placed `exec` should be sufficient.

disk_quota

Use the `disk_quota` attribute to allocate the disk space for your app instance. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case.

```yaml
---
disk_quota: 1024M
```
The command line option that overrides this attribute is `-k`.

**docker**

If your app is contained in a Docker image, then you may use the `docker` attribute to specify it and an optional Docker username.

This attribute is a combination of `push` options that include `--docker-image` and `--docker-username`.

```yaml
---
...
  docker:
    image: docker-image-repository/docker-image-name
    username: docker-user-name
---
```

The command line option `--docker-image` or `-i` overrides `docker.image`. The command line option `--docker-username` overrides `docker.username`.

The manifest attribute `docker.username` is optional. If it is used, then the password must be provided in the environment variable `CF_DOCKER_PASSWORD`. Additionally, if a Docker username is specified, then a Docker image must also be specified.

**Note:** Using the `docker` attribute in conjunction with the `buildpacks` or `path` attributes will result in an error.

**health-check-http-endpoint**

Use the `health-check-http-endpoint` attribute to customize the endpoint for the `http` health check type. If you do not provide a `health-check-http-endpoint` attribute, it uses endpoint `/`.

```yaml
---
...
  health-check-type: http
  health-check-http-endpoint: /health
---
```

**health-check-type**

Use the `health-check-type` attribute to set the `health_check_type` flag to either `port`, `process` or `http`. If you do not provide a `health-check-type` attribute, it defaults to `port`.

```yaml
---
...
  health-check-type: port
---
```

The command line option that overrides this attribute is `-u`.

The value of `none` is deprecated in favor of `process`.

**memory**

Use the `memory` attribute to specify the memory limit for all instances of an app. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case. For example:

```yaml
---
...
  memory: 1024M
---
```

The default memory limit is 1G. You might want to specify a smaller limit to conserve quota space if you know that your app instances do not require 1G of memory.

The command line option that overrides this attribute is `-m`.
no-route

By default, `cf push` assigns a route to every app. But, some apps process data while running in the background and should not be assigned routes.

You can use the `no-route` attribute with a value of `true` to prevent a route from being created for your app.

```yaml
---
  ...
  no-route: true
```

The command line option that overrides this attribute is `-no-route`.

In the Diego architecture, `no-route` skips creating and binding a route for the app, but does not specify which type of health check to perform. If your app does not listen on a port because it is a worker or a scheduler app, then it does not satisfy the port-based health check and Cloud Foundry marks it as crashed. To prevent this, disable the port-based health check with `cf set-health-check APP_NAME process`.

To remove a route from an existing app, perform the following steps:

1. Remove the route using the `cf unmap-route` command.
2. Push the app again with the `no-route: true` attribute in the manifest or the `--no-route` command line option.

For more information, see [Describing Multiple Apps with One Manifest](https://www.pivotal.io/cloudfoundry/docs/diego-describe-multiple-apps-manifest.html) below.

path

You can use the `path` attribute to tell Cloud Foundry the directory location where it can find your app.

The directory specified as the `path`, either as an attribute or as a parameter on the command line, becomes the location where the buildpack `Detect` script executes.

The command line option that overrides this attribute is `-p`.

```yaml
---
  ...
  path: /path/to/app/bits
```

For more information, see the [How cf push Finds the App](https://www.pivotal.io/cloudfoundry/docs/diego-describe-how-cf-push-finds-app.html) topic.

random-route

If you push your app without specifying any route-related CLI options or app manifest flags, the cf CLI attempts to generate a route based on the app name, which can cause collisions.

You can use the `random-route` attribute to generate a unique route and avoid name collisions. When you use `random-route`, the cf CLI generates an HTTP route with a random host (if `host` is not set) or a TCP route with an unused port number.

See the following example use cases:

- You deploy the same app to multiple spaces for testing purposes. In this situation, you can use `random-route` to randomize routes declared with the `route` attribute in the app manifest.
- You use an app manifest for a classroom training exercise in which multiple users deploy the same app to the same space.

The command line option that overrides this attribute is `--random-route`.

```yaml
---
  ...
  random-route: true
```

routes
Use the `routes` attribute to provide multiple HTTP and TCP routes. Each route for this app is created if it does not already exist.

This attribute is a combination of `push` options that include `--hostname`, `d`, and `--route-path`.

```yaml
---
...  
  routes:
    - route: example.com
    - route: www.example.com/foo
    - route: tcp-example.com:1234
---
```

**Manifest Attributes**

The `routes` attribute cannot be used in conjunction with the following attributes: `host`, `hosts`, `domain`, `domains`, and `no-hostname`. An error will result.

**Push Flag Options**

This attribute has unique interactions with different command line options.

<table>
<thead>
<tr>
<th>Push Flag Option</th>
<th>Resulting Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--no-route</code></td>
<td>All declared routes are ignored.</td>
</tr>
<tr>
<td><code>d</code></td>
<td>Overrides DOMAIN part of all declared HTTP and TCP routes.</td>
</tr>
<tr>
<td><code>--hostname</code>, <code>-n</code></td>
<td>Sets or overrides HOSTNAME in all HTTP routes. It has no impact on TCP routes.</td>
</tr>
<tr>
<td><code>--route-path</code></td>
<td>Sets or overrides the PATH in all HTTP routes. It has no impact on TCP routes.</td>
</tr>
<tr>
<td><code>--random-route</code></td>
<td>Sets or overrides the HOSTNAME in all HTTP routes. Sets or overrides the PORT in all TCP routes. The PORT and HOSTNAME will be randomly generated.</td>
</tr>
</tbody>
</table>

**stack**

Use the `stack` attribute to specify which stack to deploy your app to.

To see a list of available stacks, run `cf stacks` from the cf CLI.

```yaml
---
...  
  stack: cflinuxfs3
---
```

The command line option that overrides this attribute is `-s`.

**timeout**

The `timeout` attribute defines the number of seconds that Cloud Foundry allocates for starting your app. It is related to the `health-check-type` attribute.

For example:

```yaml
---
...  
  timeout: 80
---
```

You can increase the timeout length for very large apps that require more time to start. The `timeout` attribute defaults to 60, but you can set it to any value up to the Cloud Controller's `cc.maximum_health_check_timeout` property.

`cc.maximum_health_check_timeout` defaults to 180. On Pivotal Web Services (PWS) this is set by Pivotal and cannot be increased.

The command line option that overrides the timeout attribute is `-t`.
Environment Variables

The `env` block consists of a heading, then one or more environment variable/value pairs.

For example:

```
---
...
env:
  RAILS_ENV: production
  RACK_ENV: production
```

cf push deploys the app to a container on the server. The variables belong to the container environment.

Note: You must name variables with alphanumeric characters and underscores. Non-conforming variable names may cause unpredictable behavior.

While the app is running, you can modify environment variables.

- View all variables: `cf env my-app`
- Set an individual variable: `cf set-env my-app my-variable_name my-variable_value`
- Unset an individual variable: `cf unset-env my-app my-variable_name my-variable_value`

Environment variables interact with manifests in the following ways:

- When you deploy an app for the first time, Cloud Foundry reads the variables described in the environment block of the manifest and adds them to the environment of the container where the app is staged, and the environment of the container where the app is deployed.
- When you stop and then restart an app, its environment variables persist.

Services

Apps can bind to services such as databases, messaging, and key-value stores.

Apps are deployed into App Spaces. An app can only bind to services instances that exist in the target App Space before the app is deployed.

The `services` block consists of a heading, then one or more service instance names.

Whoever creates the service chooses the service instance names. These names can convey logical information, as in `backend_queue`, describe the nature of the service, as in `mysql_5.x`, or do neither, as in the example below.

```
---
...
services:
  - instance_ABC
  - instance_XYZ
```

Binding to a service instance is a special case of setting an environment variable, namely `VCAP_SERVICES`. See the Bind a Service section of the Delivering Service Credentials to an App topic.

Deprecated App Manifest Features

These app manifest features have been deprecated in favor of other options, as described below.

⚠️ warning: Running `cf push app -f manifest.yml` fails if your manifest uses any of these deprecated features along with the feature that replaces it.
YAML Anchors Replace Promoted Content

Previously, you could manage duplicated settings in YAML files by "promoting" the duplicate content — that is, by moving it to above the apps block, where it need appear only once.

The following example illustrates how promoted content was used to manage duplicated settings.

```yaml
---
# all apps use these settings and services
domain: shared-domain.example.com
memory: 1G
instances: 1
services:
  - clockwork-mysql
    applications:
      - name: springtock
        host: tock09876
        path: ./spring-music/build/libs/spring-music.war
      - name: springtick
        host: tick09875
        path: ./spring-music/build/libs/spring-music.war
```

Now, you can use YAML aliases instead.

The following example illustrates how to declare shared configuration using a YAML anchor, which the manifest refers to in app declarations by using an alias.

```yaml
---
defaults: &defaults
  buildpacks:
    - staticfile_buildpack
  memory: 1G

applications:
  - name: bigapp
    <<: *defaults
  - name: smallapp
    <<: *defaults
    memory: 256M
```

When pushing the app, make explicit the attributes in each app's declaration. To do this, assign the anchors and include the app-level attributes with YAML aliases in each app declaration.

Attribute routes Replaces domain, domains, host, hosts, and no-hostname

Previously, you could specify routes by listing them all at once using the `routes` attribute, or by using their hosts and domains as shown below.

```yaml
---
applications
  - name: webapp
    host: www
    domains:
      - example.com
      - example.io
```

The following route component attributes have been deprecated:

- `domain`
- `domains`
- `host`
- `hosts`
- `no-hostname`

Now you can only specify routes by using the `routes` attribute:

```yaml
---
applications
  - name: webapp
```
This app manifest feature has been deprecated, and a replacement option is under consideration.

**Inheritance**

This feature has been deprecated, and has been replaced by [Variable Substitution](#).

With inheritance, child manifests inherited configurations from a parent manifest, and the child manifests could use inherited configurations as provided, extend them, or override them. This feature has been deprecated, and has been replaced by [Variable Substitution](#).

**Buildpack Field in Manifest Is Deprecated**

The singular `buildpack` field in manifests is deprecated. It has been replaced by `buildpacks`, which is now an array which takes as a value `multiple buildpacks`.
Deploying an App with Docker

This topic describes how to use the Cloud Foundry Command Line Interface (cf CLI) to push an app with a new or updated Docker image. Cloud Foundry then uses the Docker image to create containers for the app.

See Using Docker in Cloud Foundry in the Cloud Foundry documentation for an explanation of how Docker works in Cloud Foundry.

Requirements

To push apps with Docker, you need the following:

- A Cloud Foundry deployment that has Docker support enabled. To enable Docker support, see the Enable Docker section of Using Docker in Cloud Foundry in the Cloud Foundry documentation.
- A Docker image that meets the following requirements:
  - The Docker image must contain an `/etc/passwd` file with an entry for the `root` user. In addition, the home directory and the shell for that `root` user must be present in the image file system.
  - The total size of the Docker image file system layers must not exceed the disk quota for the app. The maximum disk allocation for apps is set by the Cloud Controller. The default maximum disk quota is 2048 MB per app.
    
    Note: If the total size of the Docker image file system layers exceeds the disk quota, the app instances do not start.
  - The location of the Docker image on Docker Hub or another Docker registry.
  - A registry that supports the Docker Registry HTTP API V2 and presents a valid certificate for HTTPS traffic.

Requirement for cf ssh Support

If you want to log in to your app container using the `cf ssh` command, you must make a shell such as `sh` or `bash` available in the container.

The SSH server in the container looks for the following executables in absolute locations or the `PATH` environment variable:

- `/bin/bash`
- `/usr/local/bin/bash`
- `/bin/sh`
- `bash`
- `sh`

Benefits of Specifying Tags

If you want your app container to be consistent after platform updates and code changes, specify a tag when you push your Docker image. Otherwise, the platform applies the `latest` tag without respecting changes to `PORT` or `ENTRYPOINT`.

If you push your Docker image without specifying a tag, you must run `cf restage` for the changes to take effect.

Port Configuration

By default, apps listen for connections on the port specified in the `PORT` environment variable for the app. Cloud Foundry allocates this value dynamically.

When configuring a Docker image for Cloud Foundry, you can control the exposed port and the corresponding value of `PORT` by specifying the `EXPOSE` directive in the image Dockerfile. If you specify the `EXPOSE` directive, then the corresponding app pushed to Cloud Foundry listens on that exposed port. For example, if you set `EXPOSE 7070` to `7070`, then the app listens for connections on port 7070.
If you do not specify a port in the `EXPOSE` directive, then the app listens on the value of the `PORT` environment variable as determined by Cloud Foundry.

If you set the `PORT` environment variable via an `ENV` directive in a Dockerfile, Cloud Foundry overrides the value with the system-determined value.

Cloud Foundry supports only one exposed port on the image.

### Start Command

By default, Docker uses the start command specified in the Docker image. You can override the start command either by using a command-line parameter or by specifying it in a manifest file.

For more information about command-line parameters for `docker start`, see `docker start` in the Docker Documentation.

### Push a Docker Image From Docker Hub

To deploy a Docker image from a Docker Hub repository, run `cf push APP-NAME --docker-image REPO/IMAGE:TAG`. Replace the placeholder values in the command as follows:

- **APP-NAME**: The name of the app being pushed
- **REPO**: The name of the repository where the image is stored
- **IMAGE**: The name of an image from Docker Hub
- **TAG**: (Optional, recommended) The tag or version for the image

For example, the following command pushes the `my-image` image from Docker Hub to a Cloud Foundry app:

```
$ cf push my-app --docker-image cloudfoundry/my-image
```

### Push a Docker Image from a Private Registry

As an alternative to Docker Hub, you can use any Docker image registry that presents a valid certificate for HTTPS traffic, such as a company-internal Docker registry.

To deploy a Docker image using a specified Docker registry, run `cf push APP-NAME --docker-image MY-PRIVATE-REGISTRY.DOMAIN:PORT/REPO/IMAGE:TAG`. Replace the placeholder values in the command as follows:

- **APP-NAME**: The name of the app being pushed
- **MY-PRIVATE-REGISTRY.DOMAIN**: The path to the Docker registry
- **PORT**: The port where the registry serves traffic
- **REPO**: The name of the repository where the image is stored
- **IMAGE**: The name of the image being pushed
- **TAG**: (Optional, but recommended) The tag or version for the image

For example, the following command pushes the v2 version of the `my-image` image from the `internal-registry.example.com` registry on port 5000:

```
$ cf push my-app --docker-image internal-registry.example.com:5000/my-repo/my-image:v2
```

### Push a Docker Image From a Registry with Authentication

Many Docker registries control access to Docker images by authenticating with a username and password. Follow the steps below to deploy a Docker image with registry authentication:
1. Make sure the `CF_DOCKER_PASSWORD` environment variable is set to the Docker registry user password.

2. Run the following command:

   ```
   $ CF_DOCKER_PASSWORD=YOUR-PASSWORD cf push APP-NAME --docker-image REPO/IMAGE:TAG --docker-username USER
   ```

   Replace the placeholder values in the command as follows:

   - `YOUR-PASSWORD`: The password to use for authentication with the registry
   - `APP-NAME`: The name of the app being pushed
   - `REPO`: The repository where the image is stored:
     - For Docker Hub, this is just the repository name
     - For a private registry, this includes the registry address and port, as described in [Push a Docker Image from a Private Registry](#), in the format: `MY-PRIVATE-REGISTRY.DOMAIN:PORT/REPO`
   - `IMAGE`: The name of the image being pushed
   - `TAG`: (Optional, but recommended) The tag or version for the image
   - `USER`: The username to use for authentication with the registry

---

**Push a Docker Image From Google Container Registry (GCR)**

PCF supports pushing apps from images hosted on Google Container Registry (GCR) service. This feature requires that you use [json_key based authentication](#).

**Step 1: Authenticate with GCR**

To authenticate with GCR using PCF, you must create a JSON key file and associate it with your project:

1. Create a GCP service account. See [Creating and Enabling Service Accounts for Instances](#).

   ```
   $ gcloud iam service-accounts create MY-ACCOUNT --display-name "MY DISPLAY NAME"
   ```

2. Create a JSON key file and associate it with the service account:

   ```
   $ gcloud iam service-accounts keys create key.json --iam-account=MY-ACCOUNT@MY-PROJECT-ID.iam.gserviceaccount.com
   ```

3. Set your project ID:

   ```
   $ gcloud config set project MY-PROJECT-ID
   ```

4. Add the IAM policy binding for your project and service account:

   ```
   gcloud projects add-iam-policy-binding MY-PROJECT --member serviceAccount:MY-ACCOUNT@MY-PROJECT-ID.iam.gserviceaccount.com --role roles/storage.objectViewer
   ```

**Step 2: Deploy the GCP Image**

Run the following command to deploy your GCR image using the cf CLI:

```
CF_DOCKER_PASSWORD=$(cat key.json) cf push APP-NAME --docker-image docker://MY-REGISTRY-URL/MY-PROJECT/MY-IMAGE-NAME --docker-username _json_key
```

Replace the placeholder values in the command as follows:

- `APP-NAME`: The name of the app being pushed
- `MY-REGISTRY-URL`: The URL of your registry
- `MY-PROJECT`: The name of your project
- `MY-IMAGE-NAME`: The name of your image

**Note:** The `key.json` file must point to the file you created in the previous step.

**Note:** For information about specifying `MY-REGISTRY-URL`, see [Pushing and Pulling Images](#) on the Google Cloud documentation.
Docker Volume Support

You can use volume services with Docker apps. For more information about enabling volume support, see the Using an External File System (Volume Services) topic.
Deploying a Large App

This topic describes constraints and recommended settings for deploying apps above 750 MB.

Deployment Considerations and Limitations

The deployment process involves uploading, staging, and starting the app. See the Deployment section of the App Container Lifecycle topic for more information about the default time limits for uploading, staging, and starting an app.

To deploy large apps to PAS, ensure the following:

- The total size of the files to upload for your app does not exceed the maximum app file size that an admin sets in Ops Manager > PAS > Application Developer Controls.
- Your network connection speed is sufficient to upload your app within the 15 minute limit. We recommend a minimum speed of 874 KB/s.
- You allocate enough memory for all instances of your app. Use either the `-m` flag with `cf push` or set an app memory value in your `manifest.yml` file.
- You allocate enough disk space for all instances of your app. Use either the `-k` flag with `cf push` or set a disk space allocation value in your `manifest.yml` file.
- If you use an app manifest file, `manifest.yml`, be sure to specify adequate values for your app for attributes such as app memory, app start timeout, and disk space allocation.
  For more information about using manifests, refer to the Deploying with App Manifests topic.
- You push only the files that are necessary for your app.
  To meet this requirement, push only the directory for your app, and remove unneeded files or use the `.cfignore` file to specify excluded files.
- You configure Cloud Foundry Command Line Interface (cf CLI) staging, startup, and timeout settings to override settings in the manifest, as necessary.
  - `CF_STAGING_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to stage after Cloud Foundry successfully uploads and packages the app. Value set in minutes.
  - `CF_STARTUP_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to start. Value set in minutes.
  - `cf push -t TIMEOUT`: Controls the maximum time that Cloud Foundry allows to elapse between starting an app and the first healthy response from the app. When you use this flag, the cf CLI ignores any app start timeout value set in the manifest. Value set in seconds.
  For more information about using the cf CLI to deploy apps, refer to the Push section of the Getting Started with the cf CLI topic.

Default Settings and Limitations Summary Table

This table provides summary information of constraints and default settings to consider when you deploy a large app to PAS.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Package Size</td>
<td>Maximum: Set in Ops Manager &gt; PAS &gt; Application Developer Controls</td>
</tr>
<tr>
<td>Authorization Token Grace Period</td>
<td>Default: 20 minutes, minimum</td>
</tr>
<tr>
<td><code>CF_STAGING_TIMEOUT</code></td>
<td>cf CLI environment variable</td>
</tr>
<tr>
<td></td>
<td>Default: 15 minutes</td>
</tr>
<tr>
<td><code>CF_STARTUP_TIMEOUT</code></td>
<td>cf CLI environment variable</td>
</tr>
<tr>
<td></td>
<td>Default: 5 minutes</td>
</tr>
<tr>
<td><code>cf push -t TIMEOUT</code></td>
<td>App start timeout maximum</td>
</tr>
<tr>
<td></td>
<td>Default: 60 seconds</td>
</tr>
<tr>
<td>Disk Space Allocation</td>
<td>Default: 1024 MB</td>
</tr>
<tr>
<td>Internet Connection Speed</td>
<td>Recommended Minimum: 874 KB/s</td>
</tr>
</tbody>
</table>
Internet Connection Speed

Recommended Minimum: 874 KB/s
Starting, Restarting, and Restaging Apps

Page last updated:

This topic describes how to start, restart, and restage apps in Cloud Foundry.

Start Your App

To start your app, run the following command from your app root directory:

$ cf push YOUR-APP

For more information about deploying apps, see the Deploying an App topic.

Cloud Foundry determines the start command for your app from one of the three following sources:

- The `-c` command-line option in the Cloud Foundry Command Line Interface (cf CLI). See the following example:

  $ cf push YOUR-APP -c "node YOUR-APP.js"

- The `command` attribute in the app manifest. See the following example:

  command: node YOUR-APP.js

- The buildpack, which provides a start command appropriate for a particular type of app.

The source that Cloud Foundry uses depends on factors explained below.

How Cloud Foundry Determines its Default Start Command

The first time you deploy an app, `cf push` uses the buildpack start command by default. After that, `cf push` defaults to whatever start command was used for the previous push.

To override these defaults, provide the `-c` option, or the command attribute in the manifest. When you provide start commands both at the command line and in the manifest, `cf push` ignores the command in the manifest.

Forcing Cloud Foundry To Use the Buildpack Start Command

To force Cloud Foundry to use the buildpack start command, specify a start command of `null`.

You can specify a null start command in one of two ways.

- Using the `-c` command-line option in the cf CLI:

  $ cf push YOUR-APP -c "null"

- Using the `command` attribute in the app manifest:

  command: null

This can be helpful after you have deployed while providing a start command at the command line or the manifest. At this point, a command that you provided, rather than the buildpack start command, has become the default start command. In this situation, if you decide to deploy using the buildpack start command, the `null` command makes that easy.

Start Commands When Migrating a Database

Start commands are used in special ways when you migrate a database as part of an app deployment. For more information, see Services Overview.

Restart Your App
To restart your app, run the following command:

```bash
cf restart YOUR-APP
```

Restarting your app stops your app and restarts it with the already compiled droplet. A droplet is a tarball that includes:

- stack
- buildpack
- app source code

The Diego `cell` unpacks, compiles, and runs a droplet on a container.

Restart your app to refresh the app’s environment after actions such as binding a new service to the app or setting an environment variable that only the app consumes. However, if your environment variable is consumed by the buildpack in addition to the app, then you must `restage` the app for the change to take effect.

### Restage Your App

To restage your app, run the following command:

```bash
cf restage YOUR-APP
```

Restaging your app stops your app and restages it, by compiling a new droplet and starting it.

Restage your app if you have changed the environment in a way that affects your staging process, such as setting an environment variable that the buildpack consumes. The staging process has access to environment variables, so the environment can affect the contents of the droplet.

Restaging your app compiles a new droplet from your app without updating your app source. If you must update your app source, re-push your app by following the steps in the section above.
Pushing an App with Multiple Processes (Beta)

This topic describes how to push and manage apps with multiple processes.

Note: This is a beta feature. Because the CLI may change, this feature is not recommended for use in scripts until it is generally available.

About Processes

The CAPI v3 API supports using a single command to push apps that run multiple processes, such as a web app that has a UI process and a worker process. To push an app that creates multiple processes from the same codebase, you first define its processes in a Procfile. See Push an App with Multiple Processes.

For more information about processes, see the Processes section of the CAPI v3 documentation.

Push an App with Multiple Processes

You can push an app with multiple processes using a Procfile. Follow the procedure below.

1. Create a file named Procfile in the root of your app directory.

   Note: For more information about Procfiles, see the Procfiles section of the CAPI v3 documentation.

2. Add each process and its start command to the Procfile. See the following example:

   ```
   web: bundle exec rakeup config.ru -p $PORT
   worker: bundle exec rake worker:start
   ```

3. Run the v3-push command:

   ```
   cf v3-push myapp
   ```

   By default, the web process has a route and one instance. Other processes have zero instances by default.

Scale a Process

To scale an app process, run the following command:

```
cf v3-scale APP-NAME --process PROCESS-NAME -i INSTANCE-COUNT
```
$ cf v3-app myapp
Showing health and status for app myapp in org test / space test as admin...

name: myapp
requested state: started
processes: web:1/1, worker:2/2
memory usage: 256M x 1, 256M x 2
routes: myapp.cloudfoundry.example.com
stack: cflinuxfs3
buildpacks: ruby

web:1/1
state since cpu memory disk
#0 running 2017-09-25 15:43:26 PM 0.2% 18.9M of 256M 84.4M of 1G

worker:2/2
state since cpu memory disk
#0 running 2017-09-25 15:49:46 PM 0.1% 5M of 256M 73M of 1G
#1 running 2017-09-25 15:49:46 PM 0.1% 5M of 256M 73M of 1G
Running cf push Sub-Step Commands (Beta)

This topic describes how to run beta Cloud Foundry Command Line Interface (cf CLI) commands that provide granular control over the `cf push` process. These commands break down the `cf push` process into sub-steps that can run independently.

Note: This is a beta feature. Because the CLI may change, this feature is not recommended for use in scripts until it is generally available.

Overview

The cf CLI includes commands to provide granular control over app pushes. With these commands, you can choose to perform only some steps of the `cf push` process or perform specific actions between the steps normally executed as part of running `cf push`.

Here are some example use cases for the sub-step commands:

- Updating a third party system before staging an app
- Retrying failed stagings without incurring downtime
- Calling external services to report audit data during push
- Scanning a droplet before deploy
- Integrating with a change request system

To support these custom push workflows, Cloud Foundry divides apps into smaller building blocks. The following table describes these building blocks as Resources and lists the command associated with each one.

For information on using these commands, see the Example Workflows section below.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>The top-level resource that represents an app and its configuration. For more information, see Apps.</td>
<td>v3-create-app</td>
</tr>
<tr>
<td>Package</td>
<td>The source code that makes up an app. For more information, see Packages.</td>
<td>v3-create-package</td>
</tr>
<tr>
<td>Build</td>
<td>The staging process. Creating a build combines a Package with a Buildpack and builds it into an executable resource. For more information, see Builds.</td>
<td>v3-stage</td>
</tr>
<tr>
<td>Droplet</td>
<td>An executable resource that results from a Build. For more information, see Droplet.</td>
<td>v3-set-droplet</td>
</tr>
</tbody>
</table>

Example Workflows

The following sections describe example workflows for working with the `cf push` sub-step commands.

Push an App Using Sub-Step Commands

This example workflow describes how to push an app using sub-step commands instead of `cf push`.

1. Create your app with the cf CLI:

   ```shell
   cf v3-create-app MY-APP
   ```

2. From your app directory, create a package for your app.

   ```shell
   cf v3-create-package MY-APP
   ```

3. Locate and copy the `package_guid` from the output of the previous step. See the following example output:
Uploading and creating bits package for app MY-APP in org test / space test as admin...
package guid: 0dfca85a-8ed4-4f00-90d0-3ab08852dba8
OK

4. Stage the package you created:

```
   cf v3-stage MY-APP --package-guid PACKAGE-GUID
```

5. Locate and copy the droplet guid from the output of the previous step. See the following example output:

```
   Staging package for MY-APP in org test / space test as admin...
   ...
   Package staged
droplet guid: 06d53464-415a-4202-9d40-26a70373a487
   state: staged
   created: Mon 25 Sep 16:37:45 PDT 2018
```

6. Assign the droplet to your app:

```
   cf v3-set-droplet MY-APP -d DROPLET-GUID
```

7. Start your app:

```
   cf v3-start MY-APP
```

**Roll Back to a Previous Droplet**

This example workflow describes how to roll back to a previous droplet used by your app. You may want to use this, for example, if you update your app and it has a bug that causes it to crash.

1. List the droplets for your app:

```
   cf v3-droplets MY-APP
```

2. From the output, locate and copy the second-to-last GUID. In the following example, this is `66524145-5502-40e6-b782-476f68e13c49`.

```
   Listing droplets of app MY-APP in org test / space test as admin...
   guid state created
   66524145-5502-40e6-b782-476f68e13c49 staged Mon 25 Sep 16:37:34 PDT 2018
   0677ad93-9f77-4aaa-9a6b-44da022dcd58 staged Mon 25 Sep 16:44:55 PDT 2018
```

3. Stop your app:

```
   cf v3-stop MY-APP
```

4. Set the app to use the previous droplet:

```
   cf v3-set-droplet MY-APP -d PREVIOUS-DROPLET-GUID
```

5. Start your app:

```
   cf v3-start MY-APP
```
Rolling App Deployments (Beta)

Note: This CLI feature is currently experimental, use at your own risk. Further development of this feature will occur on CLI V7 beta.

This topic describes how developers use beta Cloud Foundry Command Line Interface (cf CLI) commands to push their apps using a rolling deployment.

For information about the traditional method for addressing app downtime while pushing app updates, see Using Blue-Green Deployment to Reduce Downtime and Risk.

Prerequisites

The procedures in this topic require the following:

- You must have cf CLI v6.40 or later.
- The rolling deployment feature must be enabled for your installation.
- Ensure that the Disable Zero Downtime App Deployments checkbox is not selected in the Advanced Features pane of the Pivotal Application Service (PAS) tile.

Commands

This section describes the beta commands for working with rolling app deployments.

Deploy an App

To deploy an app without incurring downtime, run the following command:

```
cf v3-zdt-push APP-NAME
```

Note: Ensure that you understand the Limitations of this feature before running the command.

For more information about this command, see How it Works.

Cancel a Deployment

To stop the deployment of an app that you pushed, run the following command:

```
cf v3-cancel-zdt-push APP-NAME
```

Note: The v3-cancel-zdt-push command is designed to revert the app to its original state as quickly as possible, and does not guarantee zero downtime.

This reverts the app to its state from before the deployment started by doing the following:

- Scaling up the original web process
- Removing any deployment artifacts
- Resetting the current_droplet on the app

Restart an App

Restart your app without downtime using the following command. This is useful when applying configuration updates that require a restart, such as environment variables or service bindings.
How It Works

This section describes the beta \texttt{cf v3-zdt-push} command and its limitations.

\textbf{cf v3-zdt-push Process}

This section describes the process of pushing an app with \texttt{cf v3-zdt-push}.

1. The \texttt{cf v3-zdt-push} command does the following:
   a. Stages the updated app package
   b. Creates a droplet with the updated app package
   c. Creates a deployment with the new droplet and any new configuration.

   - This starts a new process with one instance that shares the route with the old process.
   - At this point, if you run \texttt{cf app} on your app, you see a \texttt{web} process and a \texttt{web-deployment-DEPLOYMENT-UUID} process.
   - For more information about the deployment object, see the \texttt{Deployments \textasciitilde} section of the CAPI v3 docs.

2. Once the command creates the deployment, the \texttt{cc_deployment_updater} BOSH job runs in the background, updating deployments as follows:
   a. Adds another instance of the new web process and removes an instance from the old web process.

   \begin{itemize}
   \item Note: This only happens if all instances of the new web process are currently running.
   \end{itemize}

   b. Repeats the above step until the new web process reaches the desired number of instances.
   c. Removes the old web process. The new web process now fully replaces the old web process.
   d. Restarts all non-web processes of the app.
   e. Sets the deployment to \texttt{DEPLOYED}.

Limitations

The following table describes the limitations to consider when using the beta \texttt{v3-zdt-push} command.

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple app versions</td>
<td>During a deployment, Cloud Foundry serves both the old and new version of your app at the same route. This could lead to user issues if you push backwards-incompatible API changes.</td>
</tr>
<tr>
<td>Database migrations</td>
<td>Deployments do not handle database migrations. If you are migrating an app database and the old app is not compatible with the migration, you may still have downtime.</td>
</tr>
<tr>
<td>Non-web processes</td>
<td>The \texttt{v3-zdt} commands only run web processes through the rolling update sequence described above. The commands restart worker and other non-web processes in bulk after updating all web processes. The CAPI v3 API introduces the concept of processes as runnable units of an app. Each app has a web process by default. You can specify additional processes with a \texttt{procfile}, and in some cases buildpacks create additional processes. For more information about processes, see the \texttt{Processes \textasciitilde} section of the CAPI v3 documentation.</td>
</tr>
<tr>
<td>Quotas</td>
<td>Pushing updates to your app with a \texttt{v3-zdt} command creates an extra instance of your app. If there is not enough quota available, the deployment does not work. Administrators may need to increase quotas to account for rolling deployments.</td>
</tr>
<tr>
<td>Simultaneous apps when interrupting a push</td>
<td>If you run \texttt{v3-zdt-push} for an app before your previous \texttt{v3-zdt-push} command for the same app has completed, your first push gets interrupted. Until the last deployment completes, there may be many versions of the app running at once. Eventually, the app runs the code from your most recent push.</td>
</tr>
<tr>
<td>SSH to app instances</td>
<td>Pushing updates to your app with a \texttt{v3-zdt} command causes the new web process and app GUID to mismatch. \texttt{cf ssh} does not handle this scenario. You must use the \texttt{cf v3-ssh} command instead.</td>
</tr>
</tbody>
</table>
Using Blue-Green Deployment to Reduce Downtime and Risk

Blue-green deployment is a technique that reduces downtime and risk by running two identical production environments called Blue and Green.

At any time, only one of the environments is live, with the live environment serving all production traffic. For this example, Blue is currently live and Green is idle.

As you prepare a new version of your software, deployment and the final stage of testing takes place in the environment that is not live: in this example, Green. Once you have deployed and fully tested the software in Green, you switch the router so all incoming requests now go to Green instead of Blue. Green is now live, and Blue is idle.

This technique can eliminate downtime due to app deployment. In addition, blue-green deployment reduces risk: if something unexpected happens with your new version on Green, you can immediately roll back to the last version by switching back to Blue.

Note: If your app uses a relational database, blue-green deployment can lead to discrepancies between your Green and Blue databases during an update. To maximize data integrity, configure a single database for backward and forward compatibility.

Note: You can adjust the route mapping pattern to display a static maintenance page during a maintenance window for time-consuming tasks, such as migrating a database. In this scenario, the router switches all incoming requests from Blue to Maintenance to Green.

Blue-Green Deployment with Cloud Foundry Example

For this example, we’ll start with a simple app: “demo-time.” This app is a web page that displays the words “Blue time” and the date/time on the server.

Step 1: Push an App

Use the Cloud Foundry Command Line Interface (cf CLI) to push the app. Name the app “Blue” with the subdomain “demo-time.”

```bash
cf push Blue -n demo-time
```

As shown in the graphic below:

- Blue is now running on Cloud Foundry.
- The CF Router sends all traffic for demo-time.example.com traffic to Blue.

Step 2: Update App and Push

Now make a change to the app. First, replace the word “Blue” on the web page with “Green,” then rebuild the source file for the app. Run `cf push` again, but use the name “Green” for the app and provide a different subdomain to create a temporary route:

```bash
cf push Green -n demo-time-temp
```

After this push:

- Two instances of our app are now running on Cloud Foundry: the original Blue and the updated Green.
The CF Router continues sending all traffic for `demo-time.example.com` to Blue. The router now also sends any traffic for `demo-time-temp.example.com` to Green.

Step 3: Map Original Route to Green

Now that both apps are up and running, switch the router so all incoming requests go to the Green app and the Blue app. Do this by mapping the original URL route `demo-time.example.com` to the Green app using the `cf map-route` command.

```
cf map-route Green example.com -n demo-time
```

After the `cf map-route` command:

- The CF Router continues sending traffic for `demo-time-temp.example.com` to Green.
- Within a few seconds, the CF Router begins load balancing traffic for `demo-time.example.com` between Blue and Green.

Step 4: Unmap Route to Blue

Once you verify Green is running as expected, stop routing requests to Blue using the `cf unmap-route` command:

```
cf unmap-route Blue example.com -n demo-time
```

After the `cf unmap-route` command:
The CF Router stops sending traffic to Blue. Now all traffic for demo-time.example.com is sent to Green:

Step 5: Remove Temporary Route to Green
You can now use `cf unmap-route` to remove the route `demo-time-temp.example.com` from Green. The route can be deleted using `cf delete-route` or reserved for later use. You can also decommission Blue, or keep it in case you need to roll back your changes.

Implementation
Cloud Foundry community members have written a plugin to automate the blue-green deployment process:

- BlueGreenDeploy is a plugin, written in Go, for the Cloud Foundry Command Line Interface (cf CLI) that automates a few steps involved in zero-downtime deploys.
Troubleshooting app Deployment and Health

Page last updated:

Refer to this topic for help diagnosing and resolving common issues when you deploy and run apps on Cloud Foundry.

Common Issues

The following sections describe common issues you might encounter when attempting to deploy and run your app, and possible resolutions.

cf push Times Out

If your deployment times out during the upload or staging phase, you may receive one of the following error messages:

- 504 Gateway Timeout
- Error uploading application
- Timed out waiting for async job JOB-NAME to finish

If this happens, do the following:

- **Check your network speed.** Depending on the size of your app, your `cf push` could be timing out because the upload is taking too long. We recommended an Internet connection speed of at least 768 KB/s (6 Mb/s) for uploads.

- **Make sure you are pushing only needed files.** By default, `cf push` will push all the contents of the current working directory. Make sure you are pushing only the directory for your app. If your app is too large, or if it has many small files, Cloud Foundry may time out during the upload. To reduce the number of files you are pushing, ensure that you push only the directory for your app, and remove unneeded files or use the `.cfignore` file to specify excluded files.

- **Set the CF_STAGING_TIMEOUT and CF_STARTUP_TIMEOUT environment variables.** By default your app has 15 minutes to stage and 5 minutes to start. You can increase these times by setting `CF_STAGING_TIMEOUT` and `CF_STARTUP_TIMEOUT`. Type `cf push -h` at the command line for more information.

- **If your app contains a large number of files, try pushing the app repeatedly.** Each push uploads a few more files. Eventually, all files have uploaded and the push succeeds. This is less likely to work if your app has many small files.

App Too Large

If your app is too large, you may receive one of the following error messages on `cf push`:

- 413 Request Entity Too Large:
- You have exceeded your organization's memory limit

If this happens, do the following:

- **Make sure your org has enough memory for all instances of your app.** You will not be able to use more memory than is allocated for your organization. To view the memory quota for your org, use `cf org ORG_NAME`. Your total memory usage is the sum of the memory used by all apps in all spaces within the org. Each app’s memory usage is the memory allocated to it multiplied by the number of instances. To view the memory usage of all the apps in a space, use `cf apps`.

- **Make sure your app is less than 1 GB.** By default, Cloud Foundry deploys all the contents of the current working directory. To reduce the number of files you are pushing, ensure that you push only the directory for your app, and remove unneeded files or use the `.cfignore` file to specify excluded files. The following limits apply:
  - The app files to push cannot exceed 1 GB.
  - The droplet that results from compiling those files cannot exceed 1.5 GB. Droplets are typically a third larger than the pushed files.
  - The combined size of the app files, compiled droplet, and buildpack cache cannot total more than 4 GB of space during staging.

Unable to Detect a Supported App Type

If Cloud Foundry cannot identify an appropriate buildpack for your app, you will see an error message that states "Unable to detect a supported app type."

© Copyright Pivotal Software Inc, 2013-2019
You can view what buildpacks are available with the `cf buildpacks` command.

If you see a buildpack that you believe should support your app, refer to the buildpack documentation for details about how that buildpack detects apps it supports.

If you do not see a buildpack for your app, you may still be able to push your app with a custom buildpack using `cf push -b` with a path to your buildpack.

**App Deploy Fails**

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

If you push multiple apps using a manifest and one fails to deploy, CF does not attempt to push apps specified after the app that failed.

Common reasons deploying an app fail include the following:

- The app does not successfully bind to a service, such as a PostgreSQL or MongoDB. The service may not exist, may be down, or may reject the binding.
- The app does not have a unique URL. Refer to the troubleshooting tip App Requires Unique URL.

**App Requires Unique URL**

PAS requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

**App Fails to Start**

After `cf push` stages the app and uploads the droplet, the app may fail to start, commonly with a pattern of starting and crashing similar to the following example:

```
-----> Uploading droplet (23M)
...
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 down
...
0 of 1 instances running, 1 failing
FAILED
Start unsuccessful
```

If this happens, try the following:

- **Find the reason app is failing and modify your code.** Run `cf events APP-NAME` and `cf logs APP-NAME --recent` and look for messages similar to this:

  ```
  2018-07-20T15:53:26.00-0700	app.crash	index: 2, reason: CRASHED, coll_id: d874a0d5-9bca-4c53-95a-814dd90d55, instance: f40653c-b1ca-4ab6-6149-8dd, ...
  ```

  These messages may identify a memory or port issue. If they do, take that as a starting point when you re-examine and fix your app code.

- **Make sure your app code uses the PORT environment variable.** Your app may be failing because it is listening on the wrong port. Instead of hard coding the port on which your app listens, use the `PORT` environment variable.

  For example, this Ruby snippet assigns the port value to the listen_here variable:

  ```
  listen_here = ENV['PORT']
  ```

  For more examples specific to your app framework, see the appropriate buildpacks documentation for your app's language.

- **Make sure your app adheres to the principles of the Twelve-Factor App and Prepare to Deploy an App.** These texts explain how to prevent situations where your app builds locally but fails to build in the cloud.

- **Verify the timeout configuration of your app.** App health checks use a timeout setting when an app starts up for the first time. See Using Application Health Checks. If an app fails to start up due to health check timeout, you might see messages in the logs similar to the following:
App consumes too much memory, then crashes

An app that `cf push` has uploaded and started can crash later if it uses too much memory.

Make sure your app is not consuming more memory than it should. When you ran `cf push` and `cf scale`, that configured a limit on the amount of memory your app should use. Check your app's actual memory usage. If it exceeds the limit, modify the app to use less memory.

Routing Conflict

PAS allows multiple apps, or versions of the same app, to be mapped to the same route. This feature enables Blue-Green deployment. For more information see Using Blue-Green Deployment to Reduce Downtime and Risk.

Routing multiple apps to the same route may cause undesirable behavior in some situations by routing incoming requests randomly to one of the apps on the shared route.

If you suspect a routing conflict, run `cf routes` to check the routes in your installation.

If two apps share a route outside of a Blue-Green deploy strategy, choose one app to re-assign to a different route and follow the procedure below:

1. Run `cf unmap-route YOUR-APP-NAME OLD-ROUTE` to remove the existing route from that app.
2. Run `cf map-route YOUR-APP-NAME NEW-ROUTE` to map the app to a new, unique route.

Gather Diagnostic Information

Use the techniques in this section to gather diagnostic information and troubleshoot app deployment issues.

Examine Environment Variables

`cf push` deploys your app to a container on the server. The environment variables in the container govern your app.

You can set environment variables in a manifest created before you deploy. See Deploying with app Manifests.

You can also set an environment variable with a `cf set-env` command followed by a `cf push` command. You must run `cf push` for the variable to take effect in the container environment.

Use the `cf env` command to view the environment variables that you have set using the `cf set-env` command and the variables in the container environment.
Getting env variables for app my-app in org My-Org / space development as admin...

OK

System-Provided:

```
"VCAP_SERVICES": {
  "p-mysql-n/a": [
    {
      "credentials": {
        "uri": "postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lrra",
        "label": "p-mysql-n/a",
        "name": "p-mysql",
        "syslog_drain_url": "",
        "tags": ["postgres","postgresql","relational"]
      }
    }
  ]
}
```

User-Provided:

```
my-env-var: 100
my-drain: http://drain.example.com
```

View Logs

To view app logs streamed in real-time, use the `cf logs APP-NAME` command.

To aggregate your app logs to view log history, bind your app to a syslog drain service. For more information, see Streaming app Logs to Log Management Services.

### Note

The Diego architecture does not support the `cf files` command.

Trace Cloud Controller REST API Calls

If a command fails or produces unexpected results, re-run it with HTTP tracing enabled to view requests and responses between the Cloud Foundry Command Line Interface (cf CLI) and the Cloud Controller REST API.

For example:

- Re-run `cf push` with `-v`:
  ```
  cf push APP-NAME -v
  ```

- Re-run `cf push` while appending API request diagnostics to a log file:
  ```
  CF_TRACE=PATH-TO-TRACE.LOG cf push APP-NAME
  ```

These examples enable HTTP tracing for a single command only. To enable it for an entire shell session, set the variable first:

```bash
export CF_TRACE=true
```

```bash
export CF_TRACE=PATH-TO-TRACE.LOG
```

### Note

`CF_TRACE` is a local environment variable that modifies the behavior of the cf CLI. Do not confuse `CF_TRACE` with the variables in the container environment where your apps run.

Analyze Zipkin Trace IDs

When the Zipkin feature is enabled in Cloud Foundry, the Gorouter adds or forwards Zipkin trace IDs and span IDs to HTTP headers. For more information about what the Gorouter provides in the HTTP header, see the HTTP Headers section of the HTTP Routing topic.

After adding Zipkin HTTP headers to app logs, developers can use `cf logs APP-NAME` to correlate the trace and span ids logged by the Gorouter with the trace IDs.
ids logged by their app. To correlate trace ids for a request through multiple apps, each app must forward appropriate values for the headers with requests to other apps.

Use Troubleshooting Commands

You can investigate app deployment and health using the cf CLI.

Some cf CLI commands may return connection credentials. Remove credentials and other sensitive information from command output before you post the output to a public forum.

- **cf apps**: Returns a list of the apps deployed to the current space with deployment options, including the name, current state, number of instances, memory and disk allocations, and URLs of each app.
- **cf app APP-NAME**: Returns the health and status of each instance of a specific app in the current space, including instance ID number, current state, how long it has been running, and how much CPU, memory, and disk it is using.

  **Note**: CPU values returned by **cf app**: show the total usage of each app instance on all CPU cores on a host VM, where each core contributes 100%. For example, the CPU of a single-threaded app instance on a Diego cell with one core cannot exceed 100%, and four instances sharing the cell cannot exceed an average CPU of 25%. A multi-threaded app instance running alone on a cell with eight cores can draw up to 800% CPU.

- **cf env APP-NAME**: Returns environment variables set using **cf set-env** and variables existing in the container environment.
- **cf events APP-NAME**: Returns information about app crashes, including error codes. Shows that an app instance exited. For more detail, look in the app logs. See [https://github.com/cloudfoundry/errors](https://github.com/cloudfoundry/errors) for a list of Cloud Foundry errors.
- **cf logs APP-NAME --recent**: Dumps recent logs. See [Viewing Logs in the Command Line Interface](https://github.com/cloudfoundry/errors).
- **cf logs APP-NAME**: Returns a real-time stream of the app STDOUT and STDERR. Use Ctrl-C (^C) to exit the real-time stream.
- **cf files APP-NAME**: Lists the files in an app directory. Given a path to a file, outputs the contents of that file. Given a path to a subdirectory, lists the files within. Use this to **explore** individual logs.

  **Note**: Your app should direct its logs to STDOUT and STDERR. The **cf logs** command also returns messages from any **log4j** facility that you configure to send logs to STDOUT.

Access Apps with SSH

If you need to troubleshoot an instance of an app, you can gain SSH access to the app with the SSH proxy and daemon. See the [app SSH Overview](https://github.com/cloudfoundry/errors) topic for more information.

Send Requests to App Instance

To obtain debug data without SSH, you can make HTTP requests to a specific instance of an app by using the **X-CF-APP-INSTANCE** HTTP header. See the [App Instance Routing](https://github.com/cloudfoundry/errors) section of the [HTTP Routing](https://github.com/cloudfoundry/errors) topic for more information.
SSH for Apps and Services

The following list provides information about configuring and using SSH for apps and services:

- App SSH Overview
- Accessing Apps with SSH
- Accessing Services with SSH
App SSH Overview

This topic introduces SSH configuration for apps in your Pivotal Application Service (PAS) deployment.

If you need to troubleshoot an instance of an app, you can gain SSH access to the app using the SSH proxy and daemon.

For example, one of your app instances may be unresponsive, or the log output from the app may be inconsistent or incomplete. You can SSH into the individual VM that runs the problem instance to troubleshoot.

SSH Access Control Hierarchy

Operators, space managers, and space developers can configure SSH access for PAS, spaces, and apps as described in this table:

<table>
<thead>
<tr>
<th>User Role</th>
<th>Scope of SSH Permissions Control</th>
<th>How They Define SSH Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator</td>
<td>Entire deployment</td>
<td>Configure the deployment to allow or prohibit SSH access (one-time). For more information, see Configuring SSH Access for PCF.</td>
</tr>
<tr>
<td>Space Manager</td>
<td>Space</td>
<td>cf CLI allow-space-ssh and disallow-space-ssh commands</td>
</tr>
<tr>
<td>Space Developer</td>
<td>App</td>
<td>cf CLI enable-ssh and disable-ssh commands</td>
</tr>
</tbody>
</table>

An app is SSH-accessible only if operators, space managers, and space developers all grant SSH access at their respective levels. For example, the image below shows a deployment where:

- An operator allowed SSH access at the deployment level.
- A space manager allowed SSH access for apps running in spaces “A” and “B” but not “C.”
- A space developer enabled SSH access for apps that include “Foo,” “Bar,” and “Baz.”

As a result, apps “Foo,” “Bar,” and “Baz” accept SSH requests.
SSH Access for Apps and Spaces

Space managers and space developers can configure SSH access from the command line. The Cloud Foundry Command Line Interface (cf CLI) also includes commands to return the value of the SSH access setting. See the Accessing Apps with Diego SSH topic to use and configure SSH at both the app level and the space level.

Configuring SSH Access for Pivotal Application Service (PAS)

Pivotal Cloud Foundry deployments control SSH access to apps at the PAS level. Additionally, Cloud Foundry supports load balancing of SSH sessions with your load balancer. The Configuring SSH Access topic describes how to set SSH access for your deployment.

About SSH Access

The SSH system components include the SSH proxy and daemon, and the system also supports authentication, and load balancing of incoming SSH traffic. The Application SSH Components and Processes topic provides a conceptual overview.
Accessing Apps with SSH

This page assumes you are using cf CLI v6.13.0 or later.

The Cloud Foundry Command Line Interface (cf CLI) lets you securely log into remote host virtual machines (VMs) running Pivotal Application Service (PAS) app instances. This topic describes the commands that enable SSH access to apps, and enable, disable, and check permissions for such access.

The cf CLI looks up the `app_ssh_oauth_client` identifier in the Cloud Controller `/v2/info` endpoint, and uses this identifier to query the UAA server for an SSH authorization code. On the target VM side, the SSH proxy contacts the Cloud Controller through the `app_ssh_endpoint` listed in `/v2/info` to confirm permission for SSH access.

### App SSH Commands

<table>
<thead>
<tr>
<th>cf CLI Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf enable-ssh</td>
<td>Enable and Disable SSH Access</td>
</tr>
<tr>
<td>cf disable-ssh</td>
<td></td>
</tr>
<tr>
<td>cf allow-space-ssh</td>
<td></td>
</tr>
<tr>
<td>cf disallow-space-ssh</td>
<td></td>
</tr>
<tr>
<td>cf ssh-enabled</td>
<td>Check SSH Access Permissions</td>
</tr>
<tr>
<td>cf space-ssh-allowed</td>
<td>Securely log into an app container</td>
</tr>
<tr>
<td>cf ssh-code</td>
<td>Enable secure log in to an app container using non-CF SSH tools like <code>ssh</code>, <code>scp</code>, and <code>sftp</code></td>
</tr>
</tbody>
</table>

### Enabling and Disabling SSH Access

A cloud operator can deploy Pivotal Application Service (PAS) to either allow or prohibit app SSH across the entire deployment. For more information, see [Configuring SSH Access for PCF](#).

Within a deployment that permits SSH access to apps, Space Developers can enable or disable SSH access to individual apps, and Space Managers can enable or disable SSH access to all apps running within a space.

You must restart your app after enabling SSH access.

#### Configuring SSH Access at the App Level

**cf enable-ssh** enables SSH access to all instances of an app:

```
$ cf enable-ssh MY-AWESOME-APP
```

**cf disable-ssh** disables SSH access to all instances of an app:

```
$ cf disable-ssh MY-AWESOME-APP
```

#### Configuring SSH Access at the Space Level

**cf allow-space-ssh** allows SSH access into all apps in a space:

```
$ cf allow-space-ssh SPACE-NAME
```

**cf disallow-space-ssh** disallows SSH access into all apps in a space:

```
$ cf disallow-space-ssh SPACE-NAME
```
Checking SSH Permissions

cf ssh-enabled checks whether an app is accessible with SSH:

```
cf ssh-enabled MY-AWESOME-APP
ssh support is disabled for 'MY-AWESOME-APP'
```

cf space-ssh-allowed checks whether all apps running within a space are accessible with SSH:

```
cf space-ssh-allowed SPACE-NAME
ssh support is enabled in space 'SPACE-NAME'
```

Logging Into an App Container with cf SSH

If SSH access is allowed at the deployment, space, and app level, you can run the `cf ssh APP-NAME` command to start an interactive SSH session with a VM hosting an app. By default, the command accesses the container running the first instance of the app, the instance with index 0.

```
cf ssh MY-AWESOME-APP
```

When logged into a VM hosting an app, you can use tools like the Cloud Foundry Diego Operator Toolkit (cfdot) to run app status diagnostics. For more information, see [How to use CF Diego Operator Toolkit](#).

Common cf SSH Flags

You can tailor cf ssh commands with the following flags, most of which mimic flags for the Unix or Linux `ssh` command. Run the `cf ssh --help` command for more details.

- The `-i` flag targets a specific instance of an app. To log into the VM container hosting the third instance, `index=2`, of MY-AWESOME-APP, run:

```
cf ssh MY-AWESOME-APP -i 2
```

- The `-L` flag enables local port forwarding, binding an output port on your machine to an input port on the app VM. Pass in a local port, and your app VM port and port number, all colon delimited. You can prepend your local network interface, or use the default `localhost`.

```
```

- The `-N` flag skips returning a command prompt on the remote machine. This sets up local port forwarding if you do not need to execute commands on the host VM.

- The `--request-pseudo-tty` and `--force-pseudo-tty` flags let you run an SSH session in pseudo-tty mode rather than generate terminal line output.

SSH Session Environment

If you want the environment of your interactive SSH session to match the environment of your buildpack-based app, with the same environment variables and working directory, run the following commands after starting the session:

```
/lifecycle/shell
```

After running the above command, the value of the `VCAP_APPLICATION` environment variable differs slightly from its value in the environment of the app process, as it will not have the `host`, `instance_id`, `instance_index`, or `port` fields set. These fields are available in other environment variables, as described in the `VCAP_APPLICATION` documentation.
App SSH Access without cf CLI

In addition to `cf ssh`, you can use other SSH clients such as `ssh`, `scp`, or `sftp` to access your app, if you have SSH permissions.

Follow the steps below to securely connect to an app instance by logging in with a specially-formed username that passes information to the SSH proxy running on the host VM. For the password, use a one-time SSH authorization code generated by `cf ssh-code`.

1. Run `cf app MY-AWESOME-APP --guid` and record the GUID of your target app.

   ```
   $ cf app MY-AWESOME-APP --guid
   abcdefab-1234-5678-abcd-1234abcd1234
   ```

2. Query the `/v2/info` endpoint of the Cloud Controller in your deployment. Record the domain name and port of the `app_ssh_endpoint` field, and the `app_ssh_host_key_fingerprint` field. You will compare the `app_ssh_host_key_fingerprint` with the fingerprint returned by the SSH proxy on your target VM.

   ```
   $ cf curl /v2/info
   {
   ...
   "app_ssh_endpoint": "ssh.MY-DOMAIN.com:2222",
   "app_ssh_host_key_fingerprint": "a6:14:c0:ea:42:07:b2:f7:53:2c:00:66:00:21:6c",
   ...
   }
   ```

3. Run `cf ssh-code` to obtain a one-time authorization code that substitutes for an SSH password. You can run `cf ssh-code | pbcopy` to automatically copy the code to the clipboard.

   ```
   $ cf ssh-code
   E1x89n
   ```

4. Run your `ssh` or other command to connect to the app instance. For the username, use a string of the form `cf:APP-GUID/APP-INSTANCE-INDEX@SSH-ENDPOINT`, where `APP-GUID` and `SSH-ENDPOINT` come from the previous steps. For the port number, use the `SSH-PORT` recorded above. `APP-INSTANCE-INDEX` is the index of the instance you want to access.

   With the above example, you `ssh` into the container hosting the first instance of your app by running the following command:

   ```
   $ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.MY-DOMAIN.com
   ```

   Or you can use `scp` to transfer files by running one of the following commands:

   ```
   - scp -P 2222 -o User=cf:abcdefab-1234-5678-abcd-1234abcd1234/0 ssh.MY-DOMAIN.com:REMOTE-FILE-TO-RETRIEVE LOCAL-FILE-DESTINATION
   - scp -P 2222 -o User=cf:abcdefab-1234-5678-abcd-1234abcd1234/0 LOCAL-FILE-TO-COPY ssh.MY-DOMAIN.com:REMOTE-FILE-DESTINATION
   ```

5. When the SSH proxy reports its RSA fingerprint, confirm that it matches the `app_ssh_host_key_fingerprint` recorded above. When prompted for a password, paste in the authorization code returned by `cf ssh-code`.

   ```
   $ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.MY-DOMAIN.com
   The authenticity of host "ssh.MY-DOMAIN.com" (203.0.113.5) can't be established.
   RSA key fingerprint is a6:14:c0:ea:42:07:b2:f7:53:2c:00:66:00:21:6c.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added "ssh.MY-DOMAIN.com" (203.0.113.5) (RSA) to the list of known hosts.
   ```

You have now securely connected to the app instance.

### SSH Proxy Security Configuration

The SSH proxy has the following SSH security configuration by default:

<table>
<thead>
<tr>
<th>Security Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:chacha20-poly1305@openssh.com">chacha20-poly1305@openssh.com</a></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:aes128-gcm@openssh.com">aes128-gcm@openssh.com</a></td>
<td></td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
The `cf ssh` command is compatible with this security configuration. If you use a different SSH client to access apps over SSH, you should ensure that you configure your client to be compatible with these ciphers, MACs, and key exchanges. For more information about other SSH clients, see the App SSH Access without CF CLI section of this topic.

Proxy to Container Authentication

A second layer of SSH security runs within each container. When the SSH proxy attempts to handshake with the SSH daemon inside the target container, it uses the following fields associated with the `diego-ssh` key in its route to the app instance. This inner layer works invisibly and requires no user action, but is described here to complete the SSH security picture.

CONTAINER_PORT (required)

`container_port` indicates which port inside the container the SSH daemon is listening on. The proxy attempts to connect to host side mapping of this port after authenticating the client.

HOST_FINGERPRINT (optional)

When present, `host_fingerprint` declares the expected fingerprint of the SSH daemon’s host public key. When the fingerprint of the actual target’s host key does not match the expected fingerprint, the connection is terminated. The fingerprint should only contain the hex string generated by `ssh-keygen -l`.

USER (optional)

`user` declares the user ID to use during authentication with the container’s SSH daemon. While this is not a required part of the routing data, it is required for password authentication and may be required for public key authentication.

PASSWORD (optional)

`password` declares the password to use during password authentication with the container’s ssh daemon.

PRIVATE_KEY (optional)

`private_key` declares the private key to use when authenticating with the container’s SSH daemon. If present, the key must be a PEM encoded RSA or DSA public key.

Example App Process
Daemon Discovery

To be accessible via the SSH proxy, containers must host an SSH daemon, expose it via a mapped port, and advertise the port in a `diego-ssh` route. If a proxy cannot find the target process or a route, user authentication fails.

```json
"routes": {
  "diego-ssh": { "container_port": 2222 }
}
```

The Diego system generates the appropriate process definitions for Pivotal Application Service (PAS) apps which reflect the policies that are in effect.
Accessing Services with SSH

This page assumes you are using Cloud Foundry Command Line Interface (cf CLI) v6.15.0 or later.

This topic describes how to gain direct command line access to your deployed service instance. For example, you may need access to your database to execute raw SQL commands to edit the schema, import and export data, or debug app data issues.

To establish direct command line access to a service, you deploy a host app and use its SSH and port forwarding features to communicate with the service instance through the app container. The technique outlined below works with TCP services such as MySQL or Redis.

**Create a Service Instance**

1. In your terminal window, log in to your deployment with `cf login`.

2. List the marketplace services installed as product tiles on your Pivotal Cloud Foundry (PCF) Ops Manager. See the Adding and Deleting Products topic if you need to add the service as a tile. In this example, we create a p-mysql service instance.

   ```bash
cf marketplace
p-mysql 100mb MySQL databases on demand
   ``

3. Create your service instance. As part of the `create-service` command, indicate the service name, the service plan, and the name you choose for your service instance.

   ```bash
cf create-service p-mysql 100mb MY-DB
   ``

**Push Your Host App**

To push an app that will act as the host for the SSH tunnel, push any app that will successfully deploy to Pivotal Application Service (PAS).

**Note:** Your app must be prepared before you push it. See the Deploying an App topic for details on preparing apps for deployment.

1. Push your app.

   ```bash
cf push YOUR-HOST-APP
   ``

2. Enable SSH for your app.

   ```bash
cf enable-ssh YOUR-HOST-APP
   ``

   **Note:** To enable SSH access to your app, SSH access must also be enabled for both the space that contains the app and Pivotal Application Service (PAS). See the App SSH Overview topic for more details.

**Create Your Service Key**

To establish SSH access to your service instance, you must create a service key that contains critical information for configuring your SSH tunnel.

1. Create a service key for your service instance using the `cf create-service-key` command.

   ```bash
cf create-service-key MY-DB EXTERNAL-ACCESS-KEY
   ```
2. Retrieve your new service key using the `cf service-key` command.

```bash
$ cf service-key MY-DB EXTERNAL-ACCESS-KEY
Getting key EXTERNAL-ACCESS-KEY for service instance MY-DB as user@example.com
```

```json
{
  "hostname": "us-cdbr-iron-east-01.p-mysql.net",
  "jdbcUrl": "jdbc:mysql://us-cdbr-iron-east-03.p-mysql.net/ad_b2fca649704585d?user=b5136e448be920&password=231f435o05",
  "name": "ad_b2fca649704585d",
  "password": "231f435o05",
  "port": "3306",
  "uri": "mysql://b5136e448be920:231f435o05@us-cdbr-iron-east-03.p-mysql.net:3306/ad_b2fca649704585d?reconnect=true",
  "username": "b5136e448be920"
}
```

Configure Your SSH Tunnel

Configure an SSH tunnel to your service instance using `cf ssh`. Tailor the example command below with information from your service key.

```bash
$ cf ssh -L 63306:us-cdbr-iron-east-01.p-mysql.net:3306 YOUR-HOST-APP
```

- Use any available local port for port forwarding. For example, 63306.
- Replace `us-cdbr-iron-east-01.p-mysql.net` with the address provided under `hostname` in the service key retrieved above.
- Replace 3306 with the port provided under `port` above.
- Replace `YOUR-HOST-APP` with the name of your host app.

After you enter the command, open another terminal window and perform the steps below in Access Your Service Instance.

Access Your Service Instance

To establish direct command-line access to your service instance, use the relevant command line tool for that service. This example uses the MySQL command line client to access the p-mysql service instance.

```bash
$ mysql -u b5136e448be920 -h 0 -p -D ad_b2fca649704585d -P 63306
```

- Replace `b5136e448be920` with the username provided under `username` in your service key.
- `-h 0` instructs `mysql` to connect to your local machine (use `-h 127.0.0.1` for Windows).
- `-p` instructs `mysql` to prompt for a password. When prompted, use the password provided under `password` in your service key.
- Replace `ad_b2fca649704585d` with the database name provided under `name` in your service key.
- `-P 63306` instructs `mysql` to connect on port 63306.
Routes and Domains

The following list provides information about configuring routes and domains:

- Configuring Routes and Domains
- Using Weighted Routing (Beta)
- Configuring CF to Route Traffic to Apps on Custom Ports
Configuring Routes and Domains

Page last updated:

This topic describes how routes and domains work in Pivotal Application Service (PAS), and how developers and administrators configure routes and domains for their apps using the Cloud Foundry Command Line Interface (cf CLI).

For more information about routing capabilities in PAS, see the HTTP Routing topic.

Routes

The PAS Gorouter routes requests to apps by associating an app with an address, known as a route. We call this association a mapping. Use the cf CLI command to associate an app and route.

The routing tier compares each request with a list of all the routes mapped to apps and attempts to find the best match. For example, the Gorouter would make the following matches for the two routes:

<table>
<thead>
<tr>
<th>Request</th>
<th>Matched Route</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://myapp.shared-domain.example.com">http://myapp.shared-domain.example.com</a></td>
<td>myapp.shared-domain.example.com</td>
</tr>
<tr>
<td><a href="http://myapp.shared-domain.example.com/contact">http://myapp.shared-domain.example.com/contact</a></td>
<td>myapp.shared-domain.example.com</td>
</tr>
<tr>
<td><a href="http://myapp.shared-domain.example.com/products">http://myapp.shared-domain.example.com/products</a></td>
<td>myapp.shared-domain.example.com/products</td>
</tr>
<tr>
<td><a href="http://myapp.shared-domain.example.com/products/123">http://myapp.shared-domain.example.com/products/123</a></td>
<td>myapp.shared-domain.example.com/products</td>
</tr>
<tr>
<td><a href="http://products.shared-domain.example.com">http://products.shared-domain.example.com</a></td>
<td>No match; 404</td>
</tr>
</tbody>
</table>

The Gorouter does not use a route to match requests until the route is mapped to an app. In the above example, products.shared-domain.example.com may have been created as a route in Cloud Foundry, but until it is mapped to an app, requests for the route receive a 404.

The routing tier knows the location of instances for apps mapped to routes. Once the routing tier determines a route as the best match for a request, it makes a load-balancing calculation using a round-robin algorithm, and forwards the request to an instance of the mapped app.

Developers can map many apps to a single route, resulting in load-balanced requests for the route across all instances of all mapped apps. This approach enables the blue/green zero-downtime deployment strategy. Developers can also map an individual app to multiple routes, enabling access to the app from many URLs. The number of routes that can be mapped to each app is approximately 1000 (128 KB).

Routes belong to a space, and developers can only map apps to a route in the same space.

Note: Routes are globally unique. Developers in one space cannot create a route with the same URL as developers in another space, regardless of which orgs control these spaces.

HTTP vs. TCP Routes

Note: By default, PAS only supports routing of HTTP requests to apps.

Routes are considered HTTP if they are created from HTTP domains, and TCP if they are created from TCP domains. See HTTP vs. TCP Shared Domains.

HTTP routes include a domain, an optional hostname, and an optional context path. are all examples of HTTP routes. Apps should listen to the port defined by the environment variable, which is on Diego. As an example, requests to would be routed to the app container at.

Note: Developers can use the Cloud Controller API to update the ports an app can receive requests on. For more information, see Configuring Apps to Listen on Custom Ports (Beta).

- Requests to HTTP routes must be sent to ports 80 or 443.
- Ports cannot be reserved for HTTP routes.

TCP routes include a domain and a route port. A route port is the port clients make requests to. This is not the same port as what an app pushed to Cloud Foundry listens on. is an example of a TCP route. Just as for HTTP routes, apps should listen to the port defined by the environment variable, which is on Diego. As an example, requests to would be routed to the app
Once a port is reserved for a route, it cannot be reserved for another route.

Hostname and path are not supported for TCP routes.

Internal Routes

Apps can communicate without leaving the platform on the container network using internal routes. You can create an internal route using the `cf map-route` command with an internal domain.

After an internal route is mapped to an app, the route resolves to IP addresses of the app instances. The IP addresses are visible in the app container.

```
cf map-route app apps.internal --hostname app

$vcap@1234:~$ host app.apps.internal
app.apps.internal has address 10.255.169.200
app.apps.internal has address 10.255.49.7
app.apps.internal has address 10.255.49.77
```

To resolve individual instances, prepend the index to the internal route.

```
vcap@1234:~$ host 1.app.apps.internal
1.app.apps.internal has address 10.255.49.7
```

By default, apps cannot communicate with each other on the container network. To allow apps to communicate with each other you must create a network policy. For more information, see `add-network-policy` in the Cloud Foundry CLI Reference Guide.

Create a Route

When a developer creates a route using the `cf` CLI, PAS determines whether the route is an HTTP or a TCP route based on the domain. To create a HTTP route, a developer must choose an HTTP domain. To create a TCP route, a developer must choose a TCP domain.

Domains in PAS provide a namespace from which to create routes. To list available domains for a targeted organization, use the `cf domains` command. For more information about domains, see the [Domains](#) section.

The following sections describe how developers can create HTTP and TCP routes for different use cases.

Create an HTTP Route with Hostname

In PAS, a hostname is the label that indicates a subdomain of the domain associated with the route. Given a domain `shared-domain.example.com`, a developer can create the route `myapp.shared-domain.example.com` in space `my-space` by specifying the hostname `myapp` with the `cf create-route` command as shown in this example:

```
cf create-route my-space shared-domain.example.com --hostname myapp

Creating route myapp.shared-domain.example.com for org my-org / space my-space as username@example.com... OK
```

This command instructs PAS to only route requests to apps mapped to this route for the following URLs:

- `http://myapp.shared-domain.example.com`
- `https://myapp.shared-domain.example.com`
- Any path under either of the above URLs, such as `http://myapp.shared-domain.example.com/bar`

Create an HTTP Route without Hostname

This approach creates a route with the same address as the domain itself and is permitted for private domains only. For more information, see the [Private Domains](#) section.

A developer can create a route in space `my-space` from the domain `private-domain.example.com` with no hostname with the `cf create-route` command:
$ cf create-route my-space private-domain.example.com
Creating route private-domain.example.com for org my-org / space my-space as username@example.com...
OK

If DNS has been configured correctly, this command instructs PAS to route requests to apps mapped to this route from the following URLs:

- http://private-domain.example.com
- https://private-domain.example.com
- Any path under either of the above URLs, such as http://private-domain.example.com/foo

If there are no other routes for the domain, requests to any subdomain, such as http://foo.private-domain.example.com, will fail.

A developer can also create routes for subdomains with no hostnames. The following command creates a route in space my-space from the subdomain foo.private-domain.example.com:

$ cf create-route my-space foo.private-domain.example.com
Creating route foo.private-domain.example.com for org my-org / space my-space as username@example.com...
OK

Assuming DNS has been configured for this subdomain, this command instructs PAS to route requests to apps mapped to this route from the following URLs:

- http://foo.private-domain.example.com
- https://foo.private-domain.example.com
- Any path under either of the above URLs, such as http://foo.private-domain.example.com/foo

Create an HTTP Route with Wildcard Hostname

An app mapped to a wildcard route acts as a fallback app for route requests if the requested route does not exist. To create a wildcard route, use an asterisk for the hostname.

A developer can create a wildcard route in space my-space from the domain foo.shared-domain.example.com with the following command:

$ cf create-route my-space foo.shared-domain.example.com --hostname '*'
Creating route *foo.shared-domain.example.com for org my-org / space my-space as username@example.com...
OK

If a client sends a request to http://app.foo.shared-domain.example.com by accident, attempting to reach myapp.foo.shared-domain.example.com, PAS routes the request to the app mapped to the route *.foo.shared-domain.example.com.

Create an HTTP Route with a Path

Developers can use paths to route requests for the same hostname and domain to different apps.

A developer can create three routes using the same hostname and domain in the space my-space with the following commands:

$ cf create-route my-space shared-domain.example.com --hostname store --path products
Creating route store.shared-domain.example.com/products for org my-org / space my-space as username@example.com...
OK

$ cf create-route my-space shared-domain.example.com --hostname store --path orders
Creating route store.shared-domain.example.com/orders for org my-org / space my-space as username@example.com...
OK

$ cf create-route my-space shared-domain.example.com --hostname store
Creating route store.shared-domain.example.com for org my-org / space my-space as username@example.com...
OK

The developer can then map the new routes to different apps by following the steps in the Map a Route to Your App section below.

If the developer maps the first route with path products to the products app, the second route with path orders to the orders app, and the last route to the storefront app. After this, the following occurs:

- PAS routes requests to http://store.shared-domain.example.com/products to the products app.
PAS routes requests to `http://store.shared-domain.example.com/orders` to the `orders` app.

PAS routes requests to `http://store.shared-domain.example.com` to the `storefront` app.

PAS attempts to match routes with a path first, and then attempts to match host and domain.

**Note:** Routes with the same domain and hostname but different paths can only be created in the same space. Private domains do not have this limitation.

**Note:** PAS does not route requests for context paths to the root context of an app. Apps must serve requests on the context path.

Create a TCP Route with a Port

A developer can create a TCP route for `tcp.shared-domain.example.com` on an arbitrary port with the following command. If the clients of the app can accommodate addressing an arbitrary port, then developers should use the `--random-port` flag to instruct PAS to pick a port for your route.

```
cf create-route my-space tcp.shared-domain.example.com --random-port
```

Creating route tcp.shared-domain.example.com for org my-org / space my-space as user@example.com...

OK

Route tcp.shared-domain.example.com:60034 has been created

In this example, PAS routes requests to `tcp.shared-domain.example.com:60034` to apps mapped to this route.

To request a specific port, a developer can use the `--port` flag, so long as the port is not reserved for another space. The following command creates a TCP route for `tcp.shared-domain.example.com` on port 60035:

```
cf create-route my-space tcp.shared-domain.example.com --port 60035
```

Creating route tcp.shared-domain.example.com:60035 for org my-org / space my-space as user@example.com...

OK

List Routes

Developers can list routes for the current space with the `cf routes` command. A route is uniquely identified by the combination of hostname, domain, port, and path.

```
cf routes
```

```
Getting routes as user@private-domain.example.com ...

<table>
<thead>
<tr>
<th>space</th>
<th>host</th>
<th>domain</th>
<th>port</th>
<th>path</th>
<th>type</th>
<th>apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-space</td>
<td>my-app</td>
<td>shared-domain.example.com</td>
<td></td>
<td>/store</td>
<td>myapp</td>
<td>products</td>
</tr>
<tr>
<td>my-space</td>
<td>my-app</td>
<td>private-domain.example.com</td>
<td></td>
<td>/store/products</td>
<td>myapp</td>
<td>products</td>
</tr>
<tr>
<td>my-space</td>
<td>store</td>
<td>shared-domain.example.com</td>
<td>60000</td>
<td>/orders</td>
<td></td>
<td>orders</td>
</tr>
<tr>
<td>my-space</td>
<td>store</td>
<td>shared-domain.example.com</td>
<td></td>
<td>/storefront</td>
<td></td>
<td>storefront</td>
</tr>
<tr>
<td>my-space</td>
<td></td>
<td>shared-domain.example.com</td>
<td>60000</td>
<td></td>
<td>tcp</td>
<td>tcp-app</td>
</tr>
</tbody>
</table>
```

Developers can only see routes in spaces where they are members.

Check Routes

Developers cannot create a route that is already taken. To check whether a route is available, developers can use the `cf check-route` command.

The following command checks whether a route with the hostname `store` and the domain `shared-domain.example.com` and the path `products` exists:

```
cf check-route store shared-domain.example.com --path /products
```

Checking for route...  
OK  
Route store.shared-domain.example.com/products does exist

Map a Route to Your App
For an app to receive requests to a route, developers must map the route to the app with the `cf map-route` command. If the route does not already exist, this command creates it.

**Note:** Any app that is not routed to port 80 or port 443 must be explicitly mapped using the `cf map-route` command. Otherwise, the route is automatically mapped to port 443.

Developers can create and reserve routes for later use by following the steps in the [Manually Map a Route](#) section. Or they can map routes to their app immediately as part of a push by following the steps in the [Map a Route with App Push](#) section.

**Note:** Changes to route mappings are executed asynchronously. On startup, an app will be accessible at its route within a few seconds. Similarly, upon mapping a new route to a running app, the app will be accessible at this route within a few seconds of the CLI exiting successfully.

### Manually Map a Route

Given the following routes and apps:

<table>
<thead>
<tr>
<th>Route</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>store.shared-domain.example.com/products</td>
<td>products</td>
</tr>
<tr>
<td>store.shared-domain.example.com/orders</td>
<td>orders</td>
</tr>
<tr>
<td>store.shared-domain.example.com</td>
<td>storefront</td>
</tr>
<tr>
<td>tcp.shared-domain.example.com:60000</td>
<td>tcp-app</td>
</tr>
</tbody>
</table>

The following commands map the above routes to their respective apps. Developers use hostname, domain, and path to uniquely identify a route to map their apps to.

```sh
$ cf map-route products shared-domain.example.com --hostname store --path products
$ cf map-route orders shared-domain.example.com --hostname store --path orders
$ cf map-route storefront shared-domain.example.com --hostname store
$ cf map-route tcp-app tcp.shared-domain.example.com --port 60000
```

The following command maps the wildcard route `*.foo.shared-domain.example.com` to the app `myfallbackapp`.

```sh
$ cf map-route myfallbackapp foo.shared-domain.example.com --hostname "*
```

### Map a Route with App Push

Developers can map a route to their app with the `cf push` command.

If a domain or hostname is not specified, then a route will be created using the app name and the default shared domain (see [Shared Domains](#)). The following command pushes the app `myapp`, creating the route `myapp.shared-domain.example.com` from the default shared domain `shared-domain.example.com`. If the route has not already been created in another space this command also maps it to the app.

```sh
$ cf push myapp
```

To customize the route during `push`, specify the domain using the `-d` flag and the hostname with the `--hostname` flag. The following command creates the `foo.private-domain.example.com` route for `myapp`:

```sh
$ cf push myapp -d private-domain.example.com --hostname foo
```

To map a TCP route during `push`, specify a TCP domain and request a random port using `--random-route`. To specify a port, push the app without a route, then create and map the route manually by following the steps in the [Create a TCP Route with a Port](#) section.

```sh
$ cf push tcp-app -d tcp.shared-domain.example.com --random-route
```

### Map a Route Using App Manifest

© Copyright Pivotal Software Inc, 2013-2019 1165
Developers can map a route to their app with a manifest by editing the route attribute to specify the host, domain, port and/or path components of the route. For more information, see the Deploying with App Manifests topic.

Map a Route to Multiple Apps

PAS allows multiple apps, or versions of the same app, to be mapped to the same route. This feature enables Blue-Green deployment. For more information see Using Blue-Green Deployment to Reduce Downtime and Risk.

Routing multiple apps to the same route may cause undesirable behavior in some situations by routing incoming requests randomly to one of the apps on the shared route.

See the Routing Conflict section of the Troubleshooting App Deployment and Health topic for more information about troubleshooting this problem.

Map Multiple Routes to One App

You can have multiple routes to an app, but those routes cannot have different context paths.

The following routes are valid for a single app:

<table>
<thead>
<tr>
<th>Route 1</th>
<th>Route 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>myapp.example.com</td>
<td>myapp.apps.cf.example.com</td>
</tr>
<tr>
<td>myapp.example.com/foo</td>
<td>myapp.apps.cf.example.com/foo</td>
</tr>
</tbody>
</table>

The following routes are not valid for a single app:

<table>
<thead>
<tr>
<th>Route 1</th>
<th>Route 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>myapp.example.com/foo</td>
<td>myapp.apps.cf.example.com/bar</td>
</tr>
<tr>
<td>myapp.apps.cf.example.com/foo</td>
<td>myapp.example.com/bar</td>
</tr>
</tbody>
</table>

Map an Internal Route to an App

You can map an internal route to any app. This internal route allows your app to communicate with other apps without leaving the platform. Once mapped, this route becomes available to all other apps on the platform.

This example creates a foo.apps.internal internal route for myapp:

```bash
$ cf map-route myapp apps.internal --hostname foo
```

Unmap a Route

Developers can remove a route from an app using the cf unmap-route command. The route remains reserved for later use in the space where it was created until the route is deleted.

To unmap an HTTP route from an app, identify the route using the hostname, domain, and path:

```bash
$ cf unmap-route tcp-app private-domain.example.com --hostname myapp --path mypath
```

To unmap a TCP route from an app, identify the route using the domain and port:

```bash
$ cf unmap-route tcp-app tcp.shared-domain.example.com --port 60000
```

Delete a Route

Developers can delete a route from a space using the cf delete-route command.
To delete a HTTP route, identify the route using the hostname, domain, and path:

```
$ cf delete-route private-domain.example.com --hostname myapp --path mypath
```

To delete a TCP route, identify the route using the domain and port.

```
$ cf delete-route tcp.private-domain.example.com --port 60000
```

### Routing Requests to a Specific App Instance

**Note:** Usage of `X-CF-APP-INSTANCE` is supported only for Diego.

Users can route HTTP requests to a specific app instance using the header `X-CF-APP-INSTANCE`. The format of the header should be `X-CF-APP-INSTANCE:APP_GUID:APP_INDEX`. `APP_GUID` is an internal identifier for your app. Use the `cf app APP-NAME --guid` command to discover the `APP_GUID` for your app.

```
$ cf app myapp --guid
```

`APP_INDEX`, for example `0`, `1`, `2`, or `3`, is an identifier for a particular app instance. Use the CLI command `cf app APP-NAME` to get statistics on each instance of a particular app.

```
$ cf app myapp
```

The following example shows a request made to instance `9` of an app with GUID `5cd7595-2e9b-4fb2-8d5a-a66b92f2df0c` and mapped to route `myapp.private-domain.example.com`.

```
$ curl myapp.private-domain.example.com -H "X-CF-App-Instance: 5cd7595-2e9b-4fb2-8d5a-a66b92f2df0c:9"
```

If the cf CLI cannot find the instance the format is incorrect, a `404` status code is returned.

### Domains

**Note:** The term domain in this topic differs from its common use and is specific to Cloud Foundry. Likewise, shared domain and private domain refer to resources with specific meaning in Cloud Foundry. The use of domain name, root domain, and subdomain refers to DNS records.

Domains indicate to a developer that requests for any route created from the domain will be routed to PAS. This requires DNS to be configured out-of-band to resolve the domain name to the IP address of a load balancer configured to forward requests to the CF routers. For more information about configuring DNS, see the [DNS for Domains](#) section.

### List Domains for an Org

When creating a route, developers will select from domains available to them. Use the `cf domains` command to view a list of available domains for the targeted org:

```
$ cf domains
Getting domains in org my-org as user@example.com... OK
	name          status     type
shared-domain.example.com  shared
tcp.shared-domain.example.com  shared
tcp.private-domain.example.com  owned
```

This example displays three available domains: a shared HTTP domain `shared-domain.example.com`, a shared TCP domain `tcp.shared-domain.example.com`, and a private domain `tcp.private-domain.example.com`. See [Shared Domains](#) and [Private Domains](#).
HTTP vs. TCP Domains

HTTP domains indicate to a developer that only requests using the HTTP protocol will be routed to apps mapped to routes created from the domain. Routing for HTTP domains is layer 7 and offers features like custom hostnames, sticky sessions, and TLS termination.

TCP domains indicate to a developer that requests over any TCP protocol, including HTTP, will be routed to apps mapped to routes created from the domain. Routing for TCP domains is layer 4 and protocol agnostic, so many features available to HTTP routing are not available for TCP routing. TCP domains are defined as being associated with the TCP Router Group. The TCP Router Group defines the range of ports available to be reserved with TCP Routes. Currently, only Shared Domains can be TCP.

**Note:** By default, PAS only supports routing of HTTP requests to apps.

Shared Domains

Admins manage shared domains, which are available to users in all orgs of a PAS deployment. An admin can offer multiple shared domains to users. For example, an admin may offer developers the choice of creating routes for their apps from `shared-domain.example.com` and `cf.some-company.com`.

There is not technically a default shared domain. If a developer pushes an app without specifying a domain (see Map a Route with App Push), a route will be created for it from the first shared domain created in the system. All other operations involving route require the domain be specified (see Routes).

Shared domains are HTTP by default, but can be configured to be TCP when associated with the TCP Router Group.

Create a Shared Domain

Admins can create an HTTP shared domain with the `cf create-shared-domain` command:

```
$ cf create-shared-domain shared-domain.example.com
```

To create a TCP shared domain, first discover the name of the TCP Router Group.

```
$ cf router-groups
Getting router groups as admin ...
  name       type
  default-tcp tcp
```

Then create the shared domain using the `--router-group` option to associate the domain with the TCP router group.

```
$ cf create-shared-domain tcp.shared-domain.example.com --router-group default-tcp
```

Delete a Shared Domain

Admins can delete a shared domain from PAS with the `cf delete-shared-domain` command:

```
$ cf delete-shared-domain example.com
```

Internal Domain

The internal domain is a special type of shared domain used for app communication internal to the platform. When you enable service discovery, the internal domain `apps.internal` becomes available for route mapping.

Admins can configure multiple internal domains. First add a custom internal domain name to the `internal_domains` property on the `bosh-dns-adapter` job. Then create an internal domain using the `--internal` option:

```
$ cf create-shared-domain shared-domain.example.com --internal
```
The `--router-group` option is not used with internal domains.

Private Domains

Org managers can add private domains, or custom domains, and give members of the org permission to create routes for privately registered domain names. Private domains can be shared with other orgs, enabling users of those orgs to create routes from the domain.

Private domains can be HTTP or HTTPS only. TCP Routing is supported for shared domains only.

Create a Private Domain

Org managers can create a private domain with the following command:

```
$ cf create-domain my-org private-domain.example.com
```

Org managers can create a private domain for a subdomain with the following command:

```
$ cf create-domain my-org foo.private-domain.example.com
```

Sharing a Private Domain with One or More Orgs

Org managers can grant or revoke access to a private domain to other orgs if they have permissions for these orgs with the following commands:

```
$ cf share-private-domain test-org private-domain.example.com
$ cf unshare-private-domain test-org private-domain.example.com
```

Delete a Private Domain

Org managers can delete a domain from PAS with the `cf delete-domain` command:

```
$ cf delete-domain private-domain.example.com
```

Requirements for Parent and Child Domains

In the domain `myapp.shared-domain.example.com`, `shared-domain.example.com` is the parent domain of subdomain `myapp`. Note the following requirements for domains:

- You can only create a private domain that is parent to a private subdomain.
- You can create a shared domain that is parent to either a shared or a private subdomain.

The domain `foo.myapp.shared-domain.example.com` is the child subdomain of `myapp.shared-domain.example.com`. Note the following requirements for subdomains:

- You can create a private subdomain for a private parent domain only if the domains belong to the same org.
- You can create a private subdomain for a shared parent domain.
- You can only create a shared subdomain for a shared parent domain.
- You cannot create a shared subdomain for a private parent domain.

DNS for Domains

To create customized access to your apps, you can map specific or wildcard custom domains to PAS by using your DNS provider.
Mapping Domains to Your Custom Domain

To associate a registered domain name with a domain on PAS, configure a CNAME record with your DNS provider, pointing at any shared domain offered in PAS.

Mapping a Single Domain to Your Custom Domain

To map a single domain to a custom domain to PAS, configure a CNAME record with your DNS provider.

The following table provides some example CNAME record mappings.

<table>
<thead>
<tr>
<th>Record Set in Custom Domain</th>
<th>Type</th>
<th>Target in PAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>myapp.yourcustomdomain.com.</td>
<td>CNAME</td>
<td>myapp.shared-domain.example.com</td>
</tr>
<tr>
<td><a href="http://www.yourcustomdomain.com">www.yourcustomdomain.com</a>.</td>
<td>CNAME</td>
<td>myapp.shared-domain.example.com</td>
</tr>
</tbody>
</table>

After you create the CNAME mapping, your DNS provider routes your custom domain to myapp.shared-domain.example.com.

Note: Refer to your DNS provider documentation to determine whether the trailing . is required.

Mapping Multiple Subdomains to Your Custom Domain

Use a wildcard CNAME record to point all of the subdomains in your custom domain to shared-domain.example.com.

Each separately configured subdomain has priority over the wildcard configuration.

The following table provides some example wildcard CNAME record mappings.

<table>
<thead>
<tr>
<th>Record Set in Custom Domain</th>
<th>Type</th>
<th>Target in PAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.yourcustomdomain.com.</td>
<td>CNAME</td>
<td>*.shared-domain.example.com</td>
</tr>
<tr>
<td>*.yourcustomdomain.com.</td>
<td>CNAME</td>
<td>*.myapp.shared-domain.example.com</td>
</tr>
</tbody>
</table>

If you use a wildcard as the subdomain name, then your DNS provider can route from *.YOURCUSTOMDOMAIN to any of the following:

- *.shared-domain.example.com
- foo.myapp.shared-domain.example.com
- bar.foo.myapp.shared-domain.example.com

Configuring DNS for Your Registered Root Domain

To use your root domain (for example, example.com) for apps on PAS you can either use custom DNS record types like ALIAS and ANAME, if your DNS provider offers them, or subdomain redirection.

Note: Root domains are also called zone apex domains.

If your DNS provider supports using an ALIAS or ANAME record, configure your root domain with your DNS provider to point at a shared domain in PAS.

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Target</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS or ANAME</td>
<td>empty or @</td>
<td>private-domain.example.com</td>
<td>Refer to your DNS provider documentation to determine whether to use an empty or @ value for the Name entry.</td>
</tr>
</tbody>
</table>

If your DNS provider does not support ANAME or ALIAS records you can use subdomain redirection, also known as domain forwarding, to redirect requests for your root domain to a subdomain configured as a CNAME.

Note: If you use domain forwarding, SSL requests to the root domain may fail if the SSL certificate only matches the subdomain. For more information about SSL certificates, see Configuring Trusted System Certificates for Apps.

Configure the root domain to point at a subdomain such as www, and configure the subdomain as a CNAME record pointing at a shared domain in PAS.
<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Target</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL or Forward</td>
<td>private-domain.example.com</td>
<td><a href="http://www.private-domain.example.com">www.private-domain.example.com</a></td>
<td>This method results in a <a href="https://www.w3.org/Protocols/rfc2610/rfc2610.html">301 permanent redirect</a> to the subdomain you configure.</td>
</tr>
<tr>
<td>CNAME</td>
<td>www</td>
<td>myapp.shared-domain.example.com</td>
<td></td>
</tr>
</tbody>
</table>
Using Weighted Routing (Beta)

This topic provides information about weighted routing and how to configure weighted routing for your apps.

Overview

The weighted routing feature is available for Pivotal Application Service (PAS) deployments that use service mesh. For more information, see Service Mesh (Beta).

Weighted routing allows you to map multiple apps to the same route and control the amount of traffic sent to each of them. Some use cases include blue-green deployments, A/B testing, canary releases, or rolling out features over time to incremental user-bases.

How Weights are Calculated

PAS uses numeric weight values to calculate the percentage of requests that each app receives. For a route with multiple apps mapped, the percentage of requests for a given app is calculated as follows:

\[
\text{weight of app} \div \text{sum of the weights of all apps mapped to route}
\]

Consider an example in which two apps are mapped to the same route: app-a and app-b. The percentage of requests for app-b is calculated as follows:

\[
\text{weight of app-b} \div (\text{weight of app-a} + \text{weight of app-b})
\]

The following table uses example values to show how the percentage of requests is calculated for the two apps:

<table>
<thead>
<tr>
<th>App</th>
<th>Weight</th>
<th>Formula</th>
<th>Percentage of Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>app-a</td>
<td>1</td>
<td>( \frac{1}{1+3} )</td>
<td>25%</td>
</tr>
<tr>
<td>app-b</td>
<td>3</td>
<td>( \frac{3}{1+3} )</td>
<td>75%</td>
</tr>
</tbody>
</table>

In this example, you can also unmapped the route from one of the apps and observe 100% of the requests going to just one app, which simulates a migration.

Considerations for Working with Routing Weights

Consider the following when working with routing weights:

- The default weight is 1. This means that if multiple apps are mapped to the same route and their weights are not updated, traffic is equally distributed between them.
- The percentage of requests described above is not applied exactly, but over time the requests reflect the percentage.
- You cannot set a weight to 0. You can stop weighting traffic by deleting apps or unmapping apps from the route.
- You cannot exceed 100 routes mappings for a particular route. This could result in some apps not receiving any requests.

Configure Routing Weights

**Note:** This feature does not yet have cf CLI integration. The steps for updating a routing weight require you to curl the Cloud Controller v3 API.

You can configure routing weights when creating or updating a route mapping. Follow the procedure below that corresponds to your use case.

You can repeat the steps for each app that you want to receive a portion of requests at the same route.

Create a New Route Mapping

This procedure describes how to use cf to map a route to an app and assign a weight.
1. If you do not already have one, create a route. The route must use the domain created for service mesh:

   `cf create-route SPACE-NAME mesh.YOUR-APPS-DOMAIN -a HOSTNAME`

   Where:
   - **SPACE-NAME** is the space in which you want to create the route.
   - **YOUR-APPS-DOMAIN** is the shared domain for apps configured by your operator.
   - **HOSTNAME** is the hostname to use for creating the route. The format of the route is `HOSTNAME.mesh.YOUR-APPS-DOMAIN`.

2. Run the following command and record the GUID of your route:

   `cf curl /v2/routes?q=host:HOSTNAME`

   Where **HOSTNAME** is the hostname you specified when creating the route.

   The GUID of the route is located after `"metadata": { "guid":`. See the following example in which the GUID is highlighted in blue:

   ```json
   $ cf curl /v2/routes?q=host:my-hostname
   {
   "total_results": 1,
   "total_pages": 1,
   "prev_url": null,
   "next_url": null,
   "resources": [
   {
   "metadata": {
   "guid": "311d34d1-c045-4853-845f-05132377ad7d",
   "url": "/v2/routes/311d34d1-c045-4853-845f-05132377ad7d",
   "created_at": "2016-06-08T16:41:26Z",
   "updated_at": "2016-06-08T16:41:44Z"
   }
   }
   ]
   }.
   ...
   ```

3. Run the following command and record the GUID of the app that you want to map the route to:

   `cf app APP-NAME --guid`

4. Create a route mapping and assign a weight:

   ```bash
   cf curl /v3/route_mappings -X POST -d "
   { "relationships": {
   "app": { "guid": "APP-GUID" },
   "route": { "guid": "ROUTE-GUID" },
   "weight": WEIGHT
   } }
   "
   ```

   Where:
   - **APP-GUID** is the app GUID you recorded previously
   - **ROUTE-GUID** is the route GUID you recorded previously
   - **WEIGHT** is an integer used to calculate the percentage of requests the app receives. You can enter a value from 1 to 128. See How Weights are Calculated above.

**Update an Existing Route Mapping**

This procedure requires that you have mapped an app to a route in `*.mesh.YOUR-APPS-DOMAIN` either using `cf map-route`, or `cf curl` as described in Create a New Route Mapping above.

To update an existing route mapping with a new weight, do the following:

1. Run the following command and record the GUID of the app:

   `cf curl /v3/route_mappings -X POST -d "
   { "relationships": {
   "app": { "guid": "APP-GUID" },
   "route": { "guid": "ROUTE-GUID" },
   "weight": WEIGHT
   } }
   "
   ```

   Where:
   - **APP-GUID** is the app GUID you recorded previously
   - **ROUTE-GUID** is the route GUID you recorded previously
   - **WEIGHT** is an integer used to calculate the percentage of requests the app receives. You can enter a value from 1 to 128. See How Weights are Calculated above.

© Copyright Pivotal Software Inc, 2013-2019 1173 2.5
2. Run the following command and record the GUID of the route:

```
cf curl /v2/routes?q=host:HOSTNAME
```

Where `HOSTNAME` is the hostname you specified when creating the route. This is generally the same as the app name.

The GUID of the route is located after `"metadata": { "guid":`. See the following example in which the GUID is highlighted in blue:

```
$ cf curl /v2/routes?q=host:my-hostname
{
  "total_results": 1,
  "total_pages": 1,
  "prev_url": null,
  "next_url": null,
  "resources": [
    {
      "metadata": {
        "guid": "311d34d1-c045-4853-845f-05132377ad7d",
        "url": "/v2/routes/311d34d1-c045-4853-845f-05132377ad7d",
        "created_at": "2016-06-08T16:41:44Z",
        "updated_at": "2016-06-08T16:41:26Z"
      }
    }
  ]
}
```

3. List the route mappings associated with the route:

```
cf curl /v2/routes/ROUTE_GUID/route_mappings
```

Where `ROUTE_GUID` is the route GUID you got from the previous step.

4. In the output from the previous step, find the route mapping that contains the `"app_guid"` from when you ran `cf app --guid`. Record the GUID of the route mapping. See the following example output in which both the route mapping and app guids are highlighted in blue:

```
$ cf curl /v2/routes/521c375d-a7e2-4f87-9527-7fd1db1b2010/route_mappings
{
  "total_results": 1,
  "total_pages": 1,
  "prev_url": null,
  "next_url": null,
  "resources": [
    {
      "metadata": {
        "guid": "9feb9667-9249-44b7-9e4e-92157a2aaefb",
        "url": "/v2/route_mappings/9feb9667-9249-44b7-9e4e-92157a2aaefb",
        "created_at": "2016-06-08T16:41:28Z",
        "updated_at": "2016-06-08T16:41:26Z"
      }
    },
    {
      "entity": {
        "app_port": null,
        "app_guid": "bf65b03d-5416-4603-9de2-ef74491d29b6",
        "route_guid": "521c375d-a7e2-4f87-9527-7fd1db1b2010",
        "app_url": "/v2/apps/bf65b03d-5416-4603-9de2-ef74491d29b6",
        "route_url": "/v2/routes/521c375d-a7e2-4f87-9527-7fd1db1b2010"
      }
    }
  ]
}
```

5. Update the route mapping with a new weight:

```
cf curl /v3/route_mappings/ROUTE-MAPPING-GUID -X PATCH -d '{"weight": WEIGHT}'
```

Where:
- `ROUTE-MAPPING-GUID` is the route mapping GUID you got from the previous step.
- `WEIGHT` is an integer used to calculate the percentage of requests the app receives. You can enter a value from `1` to `128`. See How Weights are Calculated above.
Confirm Routing Weights

If you have two or more apps mapped to the same route, one way you can confirm routing weights is to use `curl` to view the amount of requests to each app.

To view the amount of requests to each app, do the following:

1. For each app mapped to the route, open a terminal window and run the following command. This allows you to see streaming logs for the apps.
   
   ```
   cf logs APP-NAME
   ```

2. In another terminal window, run the following command to send a `curl` request to the route every \( N \) seconds.
   
   ```
   while sleep N; do curl ROUTE; done
   ```
   
   Where:
   
   - \( N \) is the frequency, in seconds, with which you want to `curl` the route.
   - `ROUTE` is the route that you have multiple apps mapped to.

3. Observe each window in which you are running `cf logs APP-NAME`. The output displays a log message each time there is a request. The amount of requests each app receives corresponds to its configured routing weight.

4. When finished, enter \( \text{Ctrl} + \text{C} \) to stop sending requests with the `while` loop.
Configuring CF to Route Traffic to Apps on Custom Ports

This topic describes how to configure Pivotal Cloud Foundry (PCF) to route traffic to your apps listening on custom ports.

Overview

By default, apps only receive requests on port 8080 for both HTTP and TCP routing, and so must be configured, or hard coded, to listen on this port. Configuring custom app ports allows developers to bring workloads onto PAS that listen on ports other than 8080. Here are some example use cases:

- Serving web client requests on one port and offering stats/debug on another
- Using TCP protocols that require multiple ports
- Running Docker images on PAS

The Procedure below describes how to use the `apps` and `route_mappings` Cloud Controller API endpoints to update the ports the app can receive requests on.

Flow of a Request to an App

The following table describes the Network Address Translation that occurs in the data path of a client request. The procedure in this document discusses configuring app ports specifically.

<table>
<thead>
<tr>
<th>Port Type</th>
<th>Description</th>
<th>Network Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route port</td>
<td>The port a client sends a request to</td>
<td>Client to load balancer, load balancer to Gorouter</td>
</tr>
<tr>
<td>Backend port</td>
<td>The port on the VM where an app container is hosted, which is unique to the container</td>
<td>Gorouter to Diego Cell</td>
</tr>
<tr>
<td>App port</td>
<td>The port on the container; this must match a port the app is configured to listen on</td>
<td>Diego Cell to app container</td>
</tr>
</tbody>
</table>

The following diagram provides an example data path and Network Address Translation for TCP Routing. For HTTP Routing, the Route Port is always 80 or 443.

Prerequisites

Before following the procedure to configure routing to your app on custom ports, you must have the following:

- An app pushed to PAS that can listen on one or more custom ports.
- Routes for which you want traffic forwarded to your app on custom ports.
- If your app listens on two ports and you want clients to be able to send requests to both of them, create two routes. These can be from HTTP or TCP domains (see Routes and Domains for more info). Consider an example in which you have two routes: `foo.example.com` and `bar.example.com`.

In the following procedure, you use API endpoints to map these routes to your app on the ports it is listening.
Procedure

To configure your app to receive HTTP or TCP traffic on custom ports, do the following:

1. Retrieve the GUID of your app:
   
   ```bash
   cf app APP-NAME --guid
   ```
   
   Where `APP-NAME` is the name of your app.

2. Configure PAS with the ports your app is listening on:
   
   ```bash
   cf curl /v2/apps/APP-GUID -X PUT -d '{"ports": [PORT1, PORT2, PORT3...]}'
   ```
   
   Where:
   - `APP-GUID` is the GUID of your app.
   - `PORT1, PORT2, PORT3...` is a comma-separated list of the ports on which you want your app to receive traffic.

3. Retrieve the GUID of the route to which clients will make requests, and for which Cloud Foundry will route requests to the app on a custom port:
   
   ```bash
   cf curl /v2/routes?q=host:HOST-NAME
   ```
   
   Where `HOST-NAME` is the hostname for the route; by default this is the name of your app.

4. Update the route mapping for your app:
   
   ```bash
   cf curl /v2/route_mappings -X POST -d '{"app_guid": "APP-GUID", "route_guid": "ROUTE-GUID", "app_port": PORT1}.'
   ```
   
   Where:
   - `APP-GUID` is the GUID of your app.
   - `ROUTE-GUID` is the GUID of the route at which the app serves.
   - `PORT1` is the app port, or one of the app ports, that you added in the previous step.

5. Repeat the previous two steps for each port that you want your app to receive requests on.

Additional Resources

Here are additional resources related to configuring custom app ports:

- For more information about making requests to the Cloud Controller `apps` endpoint, see [Updating an App](https://github.com/cloudfoundry/community/blob/master/guides/configuration.md#updating-an-app) in the Cloud Controller API documentation.
- For more information about making requests to the Cloud Controller `route_mappings` endpoint, see [Mapping an App and a Route](https://github.com/cloudfoundry/community/blob/master/guides/configuration.md#mapping-an-app-and-a-route).
- For an example multi-port app, see the [cf-acceptance-tests](https://github.com/cloudfoundry/cf-acceptance-tests) repository.
- For a demo procedure written by an open source CF user, see “Multiple App Ports” Demo on Cloud Foundry [here](https://github.com/cloudfoundry/cf-acceptance-tests).
Managing Services

The following list provides information about managing service instances:

- Services Overview
- Managing Service Instances
- Sharing Service Instances
- Delivering Service Credentials to an App
- Managing Service Keys
- Configuring Play Framework Service Connections
- Using an External File System (Volume Services)
- User-Provided Service Instances
Managing Service Instances with the cf CLI

List Marketplace Services

Creating Service Instances

User Provided Service Instances provide a way for developers to bind apps with services that are not available in their Cloud Foundry marketplace. For more information, see the User Provided Service Instances topic.

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Example providing service-specific configuration parameters in-line:

Note: When multiple brokers provide two or more service instances with the same name, you must specify the broker by including the -b BROKER flag in the cf create-service command.
Example providing service-specific configuration parameters in a file:

```
$ cf create-service my-db-service small-plan my-db -c '{"storage_gb":4}"
```

Creating service my-db in org console / space development as user@example.com...
OK

Example providing service-specific configuration parameters in a file:

```
$ cf create-service my-db-service small-plan my-db -c /tmp/config.json
```

Creating service my-db in org console / space development as user@example.com...
OK

### Instance Tags

*Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the `VCAP_SERVICES` Environment Variable. These tags provide developers with a more generic way for apps to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including the `-t` flag followed by a comma-separated list of tags.

Example providing a comma-separated list of tags:

```
$ cf create-service my-db-service small-plan my-db -t "prod, workers"
```

Creating service my-db in org console / space development as user@example.com...
OK

### List Service Instances

Run the `cf services` command to list the service instances in your targeted space. The output from running this command includes any bound apps and the state of the last requested operation for the service instance.

```
$ cf services
Getting services in org my-org / space test as user@example.com...
OK

name     service plan    bound apps  last operation   broker
mybucket p-riakcs         developer myapp create succeeded object-store-broker
mydb     p-mysql    100mb       create succeeded mysql-broker
```

### Get Details for a Particular Service Instance

Details include dashboard urls, if applicable, and operation start and last updated timestamps.

```
$ cf service mydb
service instance: mydb
service: p-mysql
plan:    100mb
description: mysql databases on demand
documentation url: https://p-mysql.example.com/manage/instances/abcd-ef12-3456
service broker: mysql-broker

This service is not currently shared.

Showing status of last operation from service mydb...

status: create succeeded
message: started: 2019-02-13T12:02:19Z
updated: 2019-02-13T12:02:19Z

There are no bound apps for this service.
```
Bind a Service Instance

Depending on the service, you can bind service instances to apps and/or routes.

Not all services support binding, as some services deliver value to users directly without integration with Cloud Foundry, such as SaaS apps.

Bind a Service Instance to an App

Depending on the service, binding a service instance to your app may deliver credentials for the service instance to the app. See the Delivering Service Credentials to an App topic for more information. Binding a service instance to an app may also trigger app logs to be streamed to the service instance.

For more information, see Streaming App Logs to Log Management Services.

```
cf bind-service my-app mydb
Binding service mydb to my-app in org my-org / space test as user@example.com...
OK
TIP: Use `cf push` to ensure your env variable changes take effect

cf restart my-app
```

Note: You must restart or in some cases re-push your app for changes to be applied to the VCAP_SERVICES environment variable and for the app to recognize these changes.

Binding with App Manifest

As an alternative to binding a service instance to an app after pushing an app, you can use the app manifest to bind the service instance during push. As of cf CLI v6.12.1, Arbitrary Parameters are not supported in app manifests. Using the manifest to bind service instances to routes is also not supported.

The following excerpt from an app manifest binds a service instance called test-mysql-01 to the app on push.

```
services:
- test-mysql-01
```

The following excerpt from the `cf push` command and response demonstrates that the cf CLI reads the manifest and binds the service instance to an app called test-msg-app.

```
cf push
Using manifest file /Users/Bob/test-apps/test-msg-app/manifest.yml

... Binding service test-mysql-01 to test-msg-app in org My-Org / space development as user@example.com
OK
```

For more information about app manifests, see Deploying with App Manifests.

Bind a Service Instance to a Route

Binding a service instance to a route will cause app requests and responses to be proxied through the service instance, where it may be used to transform or intermediate requests. For more information, see Manage App Requests with Route Services.

```
cf bind-route-service shared-domain.example.com --hostname my-app my-service-instance
Binding route my-app.shared-domain.example.com to service instance my-service-instance in org my-org / space test as user@example.com...
OK
```

Restaging your app is not required.

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+
Some services support additional configuration parameters with the bind request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```bash
$ cf bind-service rails-sample my-db -c "{"role":"read-only"}"
Binding service my-db to app rails-sample in org console / space development as user@example.com... OK
```

```bash
$ cf bind-service rails-sample my-db -c /tmp/config.json
Binding service my-db to app rails-sample in org console / space development as user@example.com... OK
```

### Unbind a Service Instance

#### Unbind a Service Instance from an App

Unbinding a service instance from an app removes the credentials created for your app from the `VCAP_SERVICES` environment variable.

```bash
$ cf unbind-service my-app mydb
Unbinding app my-app from service mydb in org my-org / space test as user@example.com... OK
```

**Note:** You must restart or in some cases re-push your app for changes to be applied to the `VCAP_SERVICES` environment variable and for the app to recognize these changes.

#### Unbind a Service Instance from a Route

Unbinding a service instance from a route will result in requests and responses no longer being proxied through the service instance. For more information, see Manage App Requests with Route Services.

```bash
$ cf unbind-route-service shared-domain.example.com --hostname my-app my-service-instance
Unbinding may leave apps mapped to route my-app.shared-domain.example.com vulnerable; e.g. if service instance my-service-instance provides authentication. Do you want to proceed? [y] y
Unbinding route my-app.shared-domain.example.com from service instance my-service-instance in org my-org / space test as user@example.com... OK
```

Restaging your app is not required.

### Rename a Service Instance

You can change the name given to a service instance. Keep in mind that upon restarting any bound apps, the name of the instance will change in the `VCAP_SERVICES` environment variable. If your app depends on the instance name for discovering credentials, changing the name could break your app’s use of the service instance.

```bash
$ cf rename-service mydb mydb1
Renaming service mydb to mydb1 in org my-org / space test as user@example.com... OK
```

### Update a Service Instance

© Copyright Pivotal Software Inc, 2013-2019
Upgrade/Downgrade Service Plan

Changing a plan requires cf v6.7+ and cf-release v192+

By updating the service plan for an instance, users can effectively upgrade and downgrade their service instance to other service plans. Though the platform and CLI now support this feature, services must expressly implement support for it so not all services will. Further, a service might support updating between some plans but not others. For instance, a service might support updating a plan where only a logical change is required, but not where data migration is necessary. In either case, users can expect to see a meaningful error when plan update is not supported.

```
cf update-service mydb -p new-plan
Updating service instance mydb as user@example.com...
OK
```

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the update request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
cf update-service mydb -c '{"storage_gb":4}''
Updating service instance mydb as me@example.com...
```

```
cf update-service mydb -c /tmp/config.json
Updating service instance mydb as user@example.com...
```

Instance Tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the VCAP_SERVICES Environment Variable. These tags provide developers with a more generic way for apps to parse VCAP_SERVICES for credentials. Developers may provide their own tags when creating a service instance by including a comma-separated list of tags with the -t flag.

```
cf update-service my-db -t "staging, web"
Updating service my-db in org console / space development as user@example.com...
OK
```

Delete a Service Instance

Deleting a service instance deprovisions the service instance and deletes all data associated with the service instance.

```
cf delete-service mydb
Are you sure you want to delete the service mydb ? y
Deleting service mydb in org my-org / space test as user@example.com...
OK
```
Sharing Service Instances

This topic explains how to use service instance sharing.

About Service Instance Sharing

Sharing a service instance between spaces allows apps in different spaces to share databases, messaging queues, and other types of services. This eliminates the need for development teams to use service keys and user-provided services to bind their apps to the same service instance that was provisioned using the `cf create-service` command. Sharing service instances improves security, auditing, and provides a more intuitive user experience.

- Service instances can be shared into multiple spaces and across orgs.
- Developers and administrators can share service instances between spaces in which they have the Space Developer role.
- Developers who have a service instance shared with them can only bind and unbind apps to that service instance. They cannot update, rename, or delete it.
- Developers who have a service instance shared with them can view the values of any configuration parameters that were used to provision or update the service instance.

For example, if two development teams have apps in their own spaces, and both of those apps want to send messages to each other using a messaging queue, you can do the following:

1. The development team in space A can create a new instance of a messaging queue service, bind it to their app, and share that service instance into space B.
2. A developer in space B can then bind their app to the same service instance, and the two apps can begin publishing and receiving messages from one another.

Enabling Service Instance Sharing in Cloud Foundry

To enable service instance sharing, the platform operator must enable the `service_instance_sharing` flag in Cloud Foundry.

```
cf enable-feature-flag service_instance_sharing
```

Sharing a Service Instance

You can share a service instance from one space to another if you have the Space Developer role in both spaces.

To share a service instance to another space, run the following Cloud Foundry Command Line Interface (cf CLI) command:

```
cf share-service SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG]
```

- You cannot share a service instance into a space where a service instance with the same name already exists.
- To share a service instance into a space, the space must have access to the service and service plan of the service instance that you are sharing. Run the `cf enable-service-access` command to set this access.
- If you no longer have access to the service or service plan used to create your service instance, you cannot share that service instance.

Unsharing a Service Instance

⚠️ warning: Unsharing a service instance automatically deletes all bindings to apps in the spaces it was shared into. This may cause apps to fail. Before unsharing a service instance, run the `cf service SERVICE-INSTANCE` command to see how many bindings exist in the spaces the service instance is shared into.

You can unshare a service instance if you have the Space Developer role in the space where this service instance was shared from.
Developers cannot delete or rename a service instance until it is unshared from all spaces.

To unshare a service instance, run the following cf CLI command:

```
$ cf unshare-service SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG] [-f]
```

The optional `-f` flag forces unsharing without confirmation.

Security Considerations

- **Service keys** cannot be created from a space that a service instance has been shared into. This ensures that developers in the space where a service instance has been shared from have visibility into where and how many times the service instance is used.
- Sharing service instances does not automatically update app security groups (ASGs). The network policies defined in your ASGs may need to be updated to ensure that apps using shared service instances can access the underlying service.
- Access to a service must be enabled using the `cf enable-service-access` command for a service instance to be shared into a space.
- Not all services are enabled for sharing instances functionality. Contact the service vendor directly if you are unable to share instances of their service.

If you are a service author, see [Enabling Service Instance Sharing](#).

Disabling Service Instance Sharing in Cloud Foundry

To disable service instance sharing, run the following command:

```
$ cf disable-feature-flag service_instance_sharing
```

This only prevents new shares from being created. To remove existing shares, see [Deleting All Shares](#).

Deleting All Shares

The script below finds all service instances that are shared, and for each space that the service instance is shared into, all service bindings to that service instance are deleted, and all shares are deleted.

If a service binding is not successfully deleted, the script continues trying to unshare subsequent service instances.

To use this script, you must be logged in as an administrator and have jq installed.

```
#!/usr/bin/env bash

set -u
set -e

# refresh auth token
cf auth-token >/dev/null

for instance_guid in $(cf curl /v3/service_instances | jq -r '.resources[].guid'); do
    for space_guid in $(cf curl /v2/service_instances/$instance_guid/shared_to | jq -r '.resources[].space_guid'); do
        echo "Unsharing service instance $instance_guid from space $space_guid"
        set +e
        cf curl -X DELETE "/v3/service_instances/$instance_guid/relationships/shared_spaces/$space_guid"
        set -e
        done
    done
```

Note: This script has been tested on macOS Sierra 10.12.4 and Ubuntu 14.04.5. Use the script at your own risk.
Delivering Service Credentials to an App

Page last updated:

This topic describes binding apps to service instances for the purpose of generating credentials and delivering them to apps. For an overview of services, and documentation on other service management operations, see Using Services. If you are interested in building services for Cloud Foundry and making them available to end users, see the Custom Services documentation.

Bind a Service Instance

Binding a service instance to your app triggers credentials to be provisioned for the service instance and delivered to the app runtime in the VCAP_SERVICES environment variable. For details on consuming these credentials with your app, see Using Bound Service Instances.

Not all services support binding, as some services deliver value to users directly without integration with an app. In many cases binding credentials are unique to an app, and another app bound to the same service instance would receive different credentials; however this depends on the service.

```
cf bind-service my-app mydb
Binding service mydb to my-app in org my-org / space test as me@example.com... OK

TIP: Use 'cf push' to ensure your env variable changes take effect
```

```
cf restart my-app
```

**Note:** You must restart or in some cases re-push your app for changes to be applied to the VCAP_SERVICES environment variable and for the app to recognize these changes.

Arbitrary Parameters

Arbitrary parameters require Cloud Foundry Command Line Interface (cf CLI) v6.12.1 or later.

Some services support additional configuration parameters with the bind request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
cf bind-service rails-sample my-db -c '{"role":"read-only"}'
Binding service my-db to app rails-sample in org console / space development as user@example.com... OK
```

```
cf bind-service rails-sample my-db -c /tmp/config.json
Binding service my-db to app rails-sample in org console / space development as user@example.com... OK
```

Binding with App Manifest

As an alternative to binding a service instance after pushing an app, you can use the app manifest to bind the service instance during push. As of cf CLI v6.12.1, Arbitrary Parameters are not supported in app manifests.

The following excerpt from an app manifest would bind a service instance called **test-mysql-01** to the app on push.

```
services:
  - test-mysql-01
```

The following excerpt from the **cf push** command and response demonstrates that the cf CLI reads the manifest and binds the service instance to an app called **test-msg-app**.
Using manifest file /Users/Bob/test-apps/test-msg-app/manifest.yml

Binding service-test-mysql-01 to test-msg-app in org My-Org / space development as Bob@shared-domain.example.com
OK

For more information about app manifests, see Deploying with App Manifests.

Named Service Bindings

Service offering and service instance names vary across environments. App authors can discover the service instances their apps require, without having to know environment-specific service names. Developers can specify a service binding name to be included in VCAP_SERVICES.

To specify the service binding name, provide the --binding-name flag when binding an app to a service instance:

```
cf bind-service my-app my-service --binding-name postgres-database
```

The provided name will be available in the name and binding_name properties in the VCAP_SERVICES environment variable:

```
"VCAP_SERVICES": {
  "service-name": [
    {
      "name": "postgres-database",
      "binding_name": "postgres-database",
      ...
    }
  ]
}
```

Using Bound Service Instances

Once you have a service instance created and bound to your app, you need to configure the app to dynamically fetch the credentials for your service instance. The VCAP_SERVICES environment variable contains credentials and additional metadata for all bound service instances. There are two methods developers can leverage to have their apps consume binding credentials.

- **Parse the JSON yourself:** See the documentation for VCAP_SERVICES. Helper libraries are available for some frameworks.
- **Auto-configuration:** Some buildpacks create a service connection for you by creating additional environment variables, updating config files, or passing system parameters to the JVM.

For details on consuming credentials specific to your development framework, refer to the Service Binding section in the documentation for your framework’s buildpack.

Update Service Credentials

To update your service credentials, perform the following steps:

1. **Unbind the service instance** using the credentials you are updating with the following command:

   ```
cfunbind-service YOUR-APP YOUR-SERVICE-INSTANCE
```

2. **Bind the service instance** with the following command. This adds your credentials to the VCAP_SERVICES environment variable.

   ```
cfbind-service YOUR-APP YOUR-SERVICE-INSTANCE
```

3. Restart or re-push the app bound to the service instance so that the app recognizes your environment variable updates.
Unbind a Service Instance

Unbinding a service removes the credentials created for your app from the `VCAP_SERVICES` environment variable.

```
$ cf unbind-service my-app mydb
Unbinding app my-app from service mydb in org my-org / space test as me@example.com...
OK
```

**Note:** You must restart or in some cases re-push your app for changes to be applied to the `VCAP_SERVICES` environment variable and for the app to recognize these changes.
Managing Service Keys

This topic describes managing service instance credentials with service keys.

Service keys generate credentials for manually configuring consumers of marketplace services. Once you configure them for your service, local clients, apps in other spaces, or entities outside your deployment can access your service with these keys.

Note: Some service brokers do not support service keys. If you want to build a service broker that supports service keys, see Services. If you want to use a service broker that does not support service keys, see Delivering Service Credentials to an App.

Create a Service Key

To generate credentials for a service instance, use the `cf create-service-key` command:

```
$ cf create-service-key MY-SERVICE MY-KEY
Creating service key MY-KEY for service instance MY-SERVICE as me@example.com... OK
```

Use the `-c` flag to provide service-specific configuration parameters in a valid JSON object, either in-line or in a file.

To provide the JSON object in-line, use the following format:

```
$ cf create-service-key MY-SERVICE MY-KEY -c '{"read-only":true}'
Creating service key MY-KEY for service instance MY-SERVICE as me@example.com... OK
```

To provide the JSON object as a file, give the absolute or relative path to your JSON file:

```
$ cf create-service-key MY-SERVICE MY-KEY -c PATH-TO-JSON-FILE
Creating service key MY-KEY for service instance MY-SERVICE as me@example.com... OK
```

List Service Keys for a Service Instance

To list service keys for a service instance, use the `cf service-keys` command:

```
$ cf service-keys MY-SERVICE
Getting service keys for service instance MY-SERVICE as me@example.com...

  name                
  mykey1
  mykey2
```

Get Credentials for a Service Key

To retrieve credentials for a service key, use the `cf service-key` command:

```
$ cf service-key MY-SERVICE MY-KEY
Getting key MY-KEY for service instance MY-SERVICE as me@example.com...

{
  uri: foo://user2:pass2@example.com/mydb,
  servicename: mydb
}
```
Use the `--guid` flag to display the API GUID for the service key:

```bash
$ cf service-key --guid MY-SERVICE MY-KEY
Getting key MY-KEY for service instance MY-SERVICE as me@example.com...

e3696cb-7a8f-43f8-8692-43653845c7b
OK
```

**Configure Credentials for a Service Key**

Once these credentials are obtained, you can use a local CLI or utility to connect to the service instance, configure an app running outside the platform to connect to the service instance, or create a user-provided service instance so that apps in another space can connect to the service instance. How you configure these credentials will depend on what local client, app, or entity is used to access your service instance.

For more information on configuring credentials with a user-provided service instance, see [User-Provided Service Instances](#).

**Delete a Service Key**

To delete a service key, use the `cf delete-service-key` command:

```bash
$ cf delete-service-key MY-SERVICE MY-KEY
Are you sure you want to delete the service key MY-KEY? [y]
Deleting service key MY-KEY for service instance MY-SERVICE as me@example.com...
OK
```

Add option `-f` to force deletion without confirmation.

```bash
$ cf delete-service-key -f MY-SERVICE MY-KEY
Deleting service key MY-KEY for service instance MY-SERVICE as me@example.com...
OK
```
Configuring Play Framework Service Connections

Cloud Foundry provides support for connecting a Play Framework app to services such as MySQL, and Postgres. In many cases, a Play Framework app running on Cloud Foundry can automatically detect and configure connections to services.

Auto-Configuration

By default, Cloud Foundry will detect service connections in a Play Framework app and configure them to use the credentials provided in the Cloud Foundry environment. Auto-configuration will only happen if there is a single service of any of the supported types - MySQL or Postgres.
Using an External File System (Volume Services)

This topic describes how Pivotal Cloud Foundry (PCF) app developers can read and write to a mounted file system from their apps. In PCF, a volume service provides a volume so your app can read or write to a reliable, non-ephemeral file system.

Note: NFS and SMB volume services are available for Linux cells only. These services are not available for Windows cells.

Prerequisite

Before you can use a volume service with your app, your Cloud Foundry administrator must add a volume service to your deployment. See the Enabling Volume Services topic for more information.

You can run the Cloud Foundry Command Line Interface (cf CLI) command to determine if any volume services are available. See the following example output of the NFS volume service:

```
$ cf marketplace

<table>
<thead>
<tr>
<th>service</th>
<th>plans</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfs</td>
<td>Existing</td>
<td>Service for connecting to NFS volumes</td>
</tr>
</tbody>
</table>
```

If no volume service that fits your requirements exists, contact your Cloud Foundry administrator.

Mount an External Filesystem

The sections below describe how to mount an external filesystem to your app.

Create and Bind a Service Instance

To use a volume service deployed by your Cloud Foundry administrator, you must first create an instance of the specific volume service that you need. Follow the instructions below to create this service instance.

Note: For NFS-specific instructions and information, see NFS Volume Service.

1. In a terminal window, run `cf create-service SERVICE-NAME PLAN SERVICE-INSTANCE -c SHARE-JSON` to create a service instance. Replace the following with the specified values:

   - **SERVICE**: The name of the volume service that you want to use.
   - **PLAN**: The name of the service plan. Service plans are a way for providers to offer varying levels of resources or features for the same service.
   - **SERVICE-INSTANCE**: A name you provide for your service instance. Use any series of alpha-numeric characters, hyphens, and underscores. You can rename the instance at any time.
   - **SHARE-JSON** (NFS only): If you create an instance of the NFS volume service, you must supply an extra parameter, `share`, by using the `-c` flag with a JSON string, in-line or in a file. This parameter forwards information to the broker about the NFS server and share required for the service.

   The following example shows creating an instance of the “Existing” NFS service plan, passing an in-line JSON string:

   ```
   $ cf create-service nfs Existing nfs_service_instance -c '{"share": "10.10.10/10/export/myshare"}'}
   ```

2. Run `cf bind-service YOUR-APP SERVICE-NAME -c GID-AND-UID-JSON MOUNT-PATH READ-ONLY-TRUE` to bind your service instance to an app. Replace the following with the specified values:

   - **YOUR-APP**: The name of the PCF app for which you want to use the volume service.
   - **SERVICE-NAME**: The name of the volume service instance you created in the previous step.
   - **GID-AND-UID-JSON** (NFS only): If you bind an instance of the NFS volume service, you must supply two extra parameters, `gid` and `uid`. You can specify these parameters with the `-c` flag and a JSON string, in-line or from a file. This parameter specifies the `gid` and `uid` to use when mounting the share to the app.
   - **MOUNT-PATH** (Optional): To mount the volume to a particular path within your app rather than the default path, you supply the `mount` parameter. Choose a path with a root-level folder that already exists in the container, such as `/home`, `/var`, or `/var`.

Note: NFS and SMB volume services are available for Linux cells only. These services are not available for Windows cells.

© Copyright Pivotal Software Inc, 2013-2019

1192
READ-ONLY (Optional): When you issue the `cf bind-service` command, Volume Services mounts a read-write file system by default. You can specify a read-only mount by adding `"readonly":true` to the bind configuration JSON string.

The following example shows binding `my-app` to the `nfs_service_instance` and specifying a read-only volume to be mounted to `/var/volume1`, passing an in-line JSON string:

```
cf bind-service my-app nfs_service_instance -c '"uid":"1000","gid":"1000","mount":"/var/volume1","readonly":true'
```

If you use an LDAP server, you must specify `username` and `password` instead of a UID and GID in this command. For example:

```
cf bind-service my-app nfs_service_instance -c '"username":"user1000","password":"secret","mount":"/var/volume1","readonly":true'
```

3. Run `cf restage YOUR-APP` to complete the service binding by restaging your app. Replace `YOUR-APP` with the name of your app.

```
cf restage my-app
```

Access the Volume Service from your App

To access the volume service from your app, you must know which file path to use in your code. You can view the file path in the details of the service binding, which are available from the `VCAP_SERVICES` environment variable. Follow the steps below.

1. Run `cf env YOUR-APP` to view environment variables for your app. Replace `YOUR-APP` with the name of your app.

```
cf env my-app
"VCAP_SERVICES": {
    "nfs": [
    
    "credentials": {},
    "label": "nfs",
    "name": "nfs_service_instance",
    "plan": "Existing",
    "provider": null,
    "syslog_drain_url": null,
    "tags": [
      "nfs"
    ],
    "volume_mounts": [
    
    "container_dir": "/var/vcap/data/153e3c4b-1151-4cf7-b311-948dd77fc64",
    "device_type": "shared",
    "mode": "ro"
    ]
    ]
}]
```

2. Use the properties under `volume_mounts` for any information your app needs. Refer to the following table:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>container_dir</td>
<td>String containing the path to the mounted volume that you bound to your app.</td>
</tr>
<tr>
<td>device_type</td>
<td>The NFS volume release. This currently only supports shared devices. A shared device represents a distributed file system that can mount on all app instances simultaneously.</td>
</tr>
<tr>
<td>mode</td>
<td>String that informs what type of access your app has to NFS, either read-only, ro, or read and write, rw.</td>
</tr>
</tbody>
</table>

Bind with an App Manifest

You can also bind volume services using an app manifest as described in the Services section of the Deploying with App Manifests topic. However, app manifests do not support bind configuration. If you want to bind a volume service using an app manifest, you must specify bind configuration when you create the service instance. The releases that support this are `nfs-volume` v1.3.1 and later and `smb-volume` v1.0.0 and later.
Mount Path Limitation

Do not specify a `MOUNT-PATH` within the `/app` directory, which is where PAS unpacks the droplet. If you specify a mount inside the `/app` directory, the app may fail to start and parts of the app droplet may be written to the remote file share. This is because PAS mounts the volume before moving your compiled app into the droplet.

If your app requires the shared volume to be placed within the `/app` directory, do the following:

1. Specify a mount volume in a location outside of the `/app` directory.
2. Create a symbolic link at app startup time, prior to launching the app. For example:

   ```bash
   cf push my-app -c "ln -s /var/volume1 /app/volume1 && $HOME/boot.sh"
   ```

NFS Volume Service

This section describes how to use the NFS volume service.

Create an NFS Volume Service

Cloud Foundry offers two NFS volume services:

- **nfs**: This volume service provides support for NFS volumes using both v3 and v4.x protocols.
  - If you do not specify a `version` parameter in the create configuration, the protocol version used will be negotiated between client and server at mount time. This usually results in the latest available version being used.
  - If you want to request a specific protocol version, include a `version` parameter in the create configuration.
  - For read-only mounts, the driver enables attribute caching. This results in fewer attribute RPCs and better performance.
  - When you omit `uid` and `gid` or `username` and `password` in bind configuration, the driver skips `mapfs` mounting and performs just the normal kernel mount of the NFS file system without the overhead associated with FUSE mounts.

- **nfs-legacy** (deprecated): Although it is deprecated, this volume service is still available due to the difficulty of retiring services. If you use this service, it performs exactly the same mount as the `nfs` service.

Both services offer a single plan called `Existing`.

To create an NFS volume service, follow the procedure below that corresponds to your use case.

Create an `nfs` Service (auto-negotiated version)

You can create a NFS volume service using the `Existing` plan of the `nfs` service. Run the following command:

```
$ cf create-service nfs Existing SERVICE-INSTANCE-NAME -c '{"share":"SERVER/SHARE"}"
```

Where:

- **SERVICE-INSTANCE-NAME** is a name you provide for this NFS volume service instance.
- **SERVER/SHARE** is the NFS address of your server and share.

**Note**: Ensure you omit the `:` that usually follows the server name in the address.

You can run the `cf services` command to confirm that your newly-created NFS volume service displays in the output.

Create an `nfs` Service (explicit version)

You can create a NFS volume service with a specific NFS protocol version using the `Existing` plan of the `nfs` service. Run the following command:

```
$ cf create-service nfs Existing SERVICE-INSTANCE-NAME -c '{"share":"SERVER/SHARE", "version":"NFS-PROTOCOL"}"
```
Where:
- **SERVICE-INSTANCE-NAME** is a name you provide for this NFS volume service instance.
- **SERVER/SHARE** is the NFS address of your server and share.
- **VERSION** is the NFS protocol you want to use. For example, if you want to use NFSv4, set the version to 4.1. Valid values are 3.0, 4.0, 4.1 or 4.2.

**Note:** Ensure you omit the : that usually follows the server name in the address.

You can run the `cf services` command to confirm that your newly-created NFS volume service displays in the output.

**Existing **nfs-legacy** Services:**

With the release of NFS Volume Service 1.5.4, the original fuse-based NFS service has been deprecated in favor of the newer kernel-mount based NFS service and will eventually be removed. Existing NFS volume service bindings will now be listed as `nfs-legacy`. To switch over from `nfs-legacy` to the newer `nfs` service, we recommend that you re-create and re-bind your `nfs` service instances.

With the release of NFS Volume Service 2.0.0, the `nfs-legacy` service has been rewritten to use the `nfs` service. To avoid being affected when the `nfs-legacy` service is retired, re-create and re-bind your service instances using the `nfs` service.

**Deploy and Bind a Sample App**

This section describes how to deploy a sample app and bind it to the NFS volume service.

1. Clone the github repo and push the `pora` test app:
   ```
   a. cd ~/workspace
   b. git clone https://github.com/cloudfoundry/persi-acceptance-tests.git
   c. cd ~/workspace/persi-acceptance-tests/assets/pora
   d. cf push pora --no-start
   ```

2. To bind the service to your app, run the following command:
   ```
   $ cf bind-service pora SERVICE-INSTANCE-NAME -c "{"uid":"UID","gid":"GID"}"
   ```
   Where:
   - **SERVICE-INSTANCE-NAME**: The name of the volume service instance you created previously.
   - **UID** and **GID**: The gid and uid to use when mounting the share to the app. The NFS driver uses these values in the following ways:
     - When sending traffic to the NFS server, the NFS driver translates the app user id and group id to the **UID** and **GID** values.
     - When returning attributes from the NFS server, the NFS driver translates the **UID** and **GID** back to the running user uid and default gid.
   
   This allows you to interact with your NFS server as a specific user while allowing Cloud Foundry to run your app as an arbitrary user.
   - **Optional parameters**:
     - `mount`: Use this option to specify the path at which volumes mount to the app container. The default is an arbitrarily-named folder in `/var/vcap/data`. You may need to modify this value if your app has specific requirements. For example:
       ```
       cf bind-service pora myVolume -c "{"uid":"0","gid":"0","mount":"/var/path"}"
       ```
     - `readonly`: When you issue the `cf bind-service` command, Volume Services mounts a read-write file system by default. You can specify a read-only mount by adding `"readonly":true` to the bind configuration JSON string.

3. Start the app:
   ```
   $ cf start pora
   ```

4. Use the following `curl` command to confirm the app is running. The command returns an instance index for your app.
   ```
   $ curl http://pora.YOUR-CF-DOMAIN.com
   ```
5. Use the following `curl` command to confirm the app can access the shared volume. The command writes a file to the share and then reads it back out again.

```
$ curl http://pora.YOUR-CF-DOMAIN.com/write
```

**Additional Information**

This section provides additional information about using the NFS Volume Service.

**Binding with LDAP Credentials**

If your Cloud Foundry deployment has LDAP enabled, you can bind using LDAP credentials.

To bind an app to your volume using LDAP credentials, specify `username` and `password` instead of `uid` and `gid`. See the following example:

```
$ cf bind-service pora myVolume -c '{"username":"USERNAME","password":"PASSWORD"}'
```

**Note:** If your LDAP server password changes, you must re-bind your app to the service and restage. If you do not, your app will fail during restart or scaling. This is because user credentials are stored as part of the service binding and checked whenever an app is placed on a cell.

**Specifying Bind Parameters During Service Instance Creation**

As of `nfs-volume-release v1.3.1`, you can specify bind parameters in advance, when you create a service instance. Use this option if you bind the service to your app in an app manifest, where bind configuration is not supported.

**File Locking with `flock()` and `lockf()` / `fcntl()`**

Apps that use file locking through unix system calls such as `flock()` and `lockf()` or script commands such as `flock` may use the `nfs` service. The `nfs-legacy` service uses a fuse mounting process that does not enforce locks across Diego cells.

**SMB Volume Service**

This section describes how to use an SMB (Server Message Block) volume service.

**Create an SMB Volume Service**

Cloud Foundry offers one SMB volume service:

- `smb`: This volume service provides support for existing SMB shares.

The service offers a single plan called `Existing`.

To create an SMB volume service, follow the procedure below.

**Create with `smb` Service**

You can create an SMB volume service using the `Existing` plan of the `smb` service. Run the following command:

```
$ cf create-service smb Existing SERVICE-INSTANCE-NAME -c '{"share":"//SERVER/SHARE"}'
```

Where:

- `SERVICE-INSTANCE-NAME` is a name you provide for this SMB volume service instance.
* //SERVER/SHARE is the SMB address of your server and share.

You can run the `cf services` command to confirm that your newly-created SMB volume service displays in the output.

## Deploy and Bind a Sample App

This section describes how to deploy a sample app and bind it to the SMB volume service.

1. Clone the github repo and push the `pora` test app:

   a. `cd ~/workspace`
   b. `git clone https://github.com/cloudfoundry/persi-acceptance-tests.git`
   c. `cd ~/workspace/persi-acceptance-tests/assets/pora`
   d. `cf push pora --no-start`

2. To bind the service to your app, run the following command:

   ```
   $ cf bind-service pora SERVICE-INSTANCE-NAME -c '"username":"USERNAME","password":"PASSWORD"'
   ```

   Where:
   - `SERVICE-INSTANCE-NAME`: The name of the volume service instance you created previously.
   - `USERNAME` and `PASSWORD`: The username and password to use when mounting the share to the app. This allows you to interact with your SMB server as a specific user while allowing Cloud Foundry to run your app as an arbitrary user.
   - **Optional parameters:**
     - `mount`: Use this option to specify the path at which volumes mount to the app container. The default is an arbitrarily-named folder in `/var/vcap/data`. You may need to modify this value if your app has specific requirements. For example:
     ```
     $ cf bind-service pora myVolume -c '"username":"some-user","password":"some-password","mount":"/var/path"'
     ```
     - `readonly`: When you issue the `cf bind-service` command, Volume Services mounts a read-write file system by default. You can specify a read-only mount by adding `"readonly":true` to the bind configuration JSON string.
     - `domain`: In case if your user is in windows domain you can specify `domain` parameter.

3. Start the app:

   ```
   $ cf start pora
   ```

4. Use the following `curl` command to confirm the app is running. The command returns an instance index for your app.

   ```
   $ curl http://pora.YOUR-CF-DOMAIN.com
   ```

5. Use the following `curl` command to confirm the app can access the shared volume. The command writes a file to the share and then reads it back out again.

   ```
   $ curl http://pora.YOUR-CF-DOMAIN.com/write
User-Provided Service Instances

Page last updated:

This topic describes how to create and update user-provided service instances.

Note: The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also create user-provided service instances in Apps Manager from the Marketplace. To update existing user-provided service instances, navigate to the service instance page and select the Configuration tab.

Overview

User-provided service instances enable developers to use services that are not available in the marketplace with their apps running on Cloud Foundry.

User-provided service instances can be used to deliver service credentials to an app, and/or to trigger streaming of app logs to a syslog compatible consumer. These two functions can be used alone or at the same time.

Once created, user-provided service instances behave like service instances created through the marketplace; see Managing Service Instances and App Binding for details on listing, renaming, deleting, binding, and unbinding.

Create a User-Provided Service Instance

The alias for cf create-user-provided-service is cf cups.

Deliver Service Credentials to an App

Suppose a developer obtains a URL, port, username, and password for communicating with an Oracle database managed outside of Cloud Foundry. The developer could manually create custom environment variables to configure their app with these credentials (of course you would never hard code these credentials in your app!).

User-provided service instances enable developers to configure their apps with these using the familiar App Binding operation and the same app runtime environment variable used by Cloud Foundry to automatically deliver credentials for marketplace services (VCAP_SERVICES).

To create a service instance in interactive mode, use the -p option with a comma-separated list of parameter names. The Cloud Foundry Command Line Interface (cf CLI) prompts you for each parameter value.

Create the user-provided service instance, specifying the URL of the service with the -l option.

Stream App Logs to a Service

User-provided service instances enable developers to stream app logs to a syslog compatible aggregation or analytics service that isn't available in the marketplace. For more information about the syslog protocol see RFC 5424 and RFC 6587.

© Copyright Pivotal Software Inc, 2013-2019
To stream app logs to the service, bind the user-provided service instance to your app.

### Proxy App Requests to a Route Service

User-provided service instances enable developers to proxy app requests to [route services](#) for preprocessing. To create a user-provided service instance for a route service, specify the url for the route service using the `--route-service-url` option.

```bash
$ cf create-user-provided-service my-user-provided-route-service --route-service-url https://my-route-service.example.com
Creating user provided service my-user-provided-route-service in org my-org / space my-space as user@example.com... OK
```

**Note:** When creating the user-provided service, the route service url specified must be https.

To proxy requests to the user-provided route service, you must bind the service instance to the route. For more information, see [Manage App Requests with Route Services](#).

### Update a User-provided Service Instance

You can use `cf update-user-provided-service` to update the attributes of an instance of a user-provided service. New credentials overwrite old credentials, and parameters not provided are deleted.

The alias for `update-user-provided-service` is `uups`.
Streaming App Logs

The following list provides information about streaming app logs:

- Streaming App Logs to Log Management Services
- Service-Specific Instructions for Streaming App Logs
- Streaming App Logs to Splunk
- Streaming App Logs with Fluentd
- Streaming App Logs to Azure OMS Log Analytics
Streaming App Logs to Log Management Services

Page last updated:

This topic describes how to drain logs from Cloud Foundry to a third-party log management service.

Cloud Foundry aggregates logs for all instances of your apps as well as for requests made to your apps through internal components of Cloud Foundry. For example, when the Cloud Foundry Router forwards a request to an app, the Router records that event in the log stream for that app. Run the following command to access the log stream for an app in the terminal:

```
cf logs YOUR-APP-NAME
```

If you want to persist more than the limited amount of logging information that Cloud Foundry can buffer, drain these logs to a log management service.

For more information about the systems responsible for log aggregation and streaming in Cloud Foundry, see [App Logging in Cloud Foundry](#).

Using Services from the Cloud Foundry Marketplace

Your Cloud Foundry marketplace may offer one or more log management services. To use one of these services, create an instance of the service and bind it to your app with the following commands:

```
cf create-service SERVICE PLAN SERVICE-INSTANCE
cf bind-service YOUR-APP YOUR-LOG-STORE
```

For more information about service instance lifecycle management, see the [Managing Service Instances](#) topic.

**Note:** Not all marketplace services support syslog drains. Some services implement an integration with Cloud Foundry that enables automated streaming of app syslogs. If you are interested in building services for Cloud Foundry and making them available to end users, see the [Custom Services](#) documentation.

Using Services Not Available in Your Marketplace

If a compatible log management service is not available in your Cloud Foundry marketplace, you can use user-provided service instances to stream app logs to a service of your choice.

You can install and use the [CF Drain CLI Plugin](#) to create and manage user-provided syslog drains from the CF command-line interface (cf CLI).

You may need to prepare your log management service to receive app logs from Cloud Foundry. For specific instructions for several popular services, see [Service-Specific Instructions for Streaming App Logs](#). If you cannot find instructions for your service, follow the generic instructions below.

**Step 1: Configure the Log Management Service**

Complete the following steps to set up a communication channel between the log management service and your Cloud Foundry deployment:

1. Obtain the external IP addresses that your Cloud Foundry administrator assigns to outbound traffic.
2. Provide these IP addresses to the log management service. The specific steps to configure a third-party log management service depend on the service.
3. Whitelist these IP addresses to ensure unrestricted log routing to your log management service.
4. Record the syslog URL provided by the third-party service. Third-party services typically provide a syslog URL to use as an endpoint for incoming log data. You use this syslog URL in Step 2: Create a User-provided Service Instance.

Cloud Foundry uses the syslog URL to route messages to the service. The syslog URL has a scheme of [syslog](#), [syslog-tls](#), or [https](#), and can include a port number. For example:

```
syslog://logs.example.com:2734
```
Step 2: Create and Bind a User-Provided Service Instance

You can create a syslog drain service and bind apps to it using either generic Cloud Foundry Command Line Interface (cf CLI) commands, or drain-specific commands enabled by the CF Drain plugin for the cf CLI.

Each option is described below.

With the CF Drain CLI Plugin

1. If the CF Drain CLI Plugin is not installed on your local workstation, follow the [Installing Plugin](#) instructions in the plugin source repository.

2. Decide whether to bind the drain to a single app or all apps in a space, and run the corresponding command:

   - **Single app:**
     ```
cf drain APP-NAME SYSLOG-DRAIN-URL
     ```
   
     Where:
     - **APP-NAME** is the name of the app to stream logs from.
     - **SYSLOG-DRAIN-URL** is the syslog URL from [Step 1: Configure the Log Management Service](#).

   - **All apps in a space:**
     ```
cf drain-space --drain-name DRAIN-NAME --drain-url SYSLOG-DRAIN-URL --username USERNAME
     ```

     Where:
     - **DRAIN-NAME** is the name of the app to stream logs from.
     - **SYSLOG-DRAIN-URL** is the syslog URL from [Step 1: Configure the Log Management Service](#).
     - **USERNAME** is the username to use when pushing the app. If you do not specify a username, you must have admin permissions because the plugin will create a user.

After a short delay, logs begin to flow automatically.

Refer to the [Usage](#) section of the CF Drain plugin source repository for CF Drain commands, and [Managing Service Instances with the CLI](#) for general CF service commands.

With General cf CLI Service Commands

1. To create the service instance, run `cf create-user-provided-service` (or `cf cups`) with the `-l` flag, filling in values as follows:
   - **DRAIN-NAME**: A name to use for your syslog drain service instance
   - **SYSLOG-DRAIN-URL**: The syslog URL from [Step 1: Configure the Log Management Service](#).

   ```
cf create-user-provided-service DRAIN-NAME -l SYSLOG-DRAIN-URL
   ```

   See [User-Provided Service Instances](#) for more information.

2. To bind an app to the service instance, do one of the following:
   - Run `cf push` with a manifest. The services block in the manifest must specify the service instance that you want to bind.
   - Run `cf bind-service`.

   ```
cf bind-service YOUR-APP-NAME DRAIN-NAME
   ```

**Note**: To bind a drain to all apps in a space with a single command, you must use the CF Drain CLI Plugin as described in the previous section.

© Copyright Pivotal Software Inc, 2013-2019
After a short delay, logs begin to flow automatically.

Refer to Managing Service Instances with the CLI for more information.

Step 3: Verify Logs Are Draining

To verify that logs are draining correctly to a third-party log management service:

1. Take actions that produce log messages, such as making requests of your app.
2. Compare the logs displayed in the CLI against those displayed by the log management service.

For example, if your app serves web pages, you can send HTTP requests to the app. In Cloud Foundry, these generate Router log messages, which you can view in the CLI. Your third-party log management service should display corresponding messages.

Note: For security reasons, Cloud Foundry apps do not respond to `ping`. You cannot use `ping` to generate log entries.

CF Drain CLI Plugin

The CF Drain CLI plugin extends the cf CLI by adding simple commands for user-provided syslog drains. Also, you can use the plugin to bind all apps in a space to a syslog drain. This option includes app, space, and org names in the drain. It also binds any new apps pushed to the space.

Installation: To install the CF Drain CLI plugin see the Installing Plugin instructions in the plugin source repository.

Commands: The plugin adds commands for creating, deleting, and listing syslog drains, and for binding apps to the drains. See the Usage section of the plugin source repository for details.
Service-Specific Instructions for Streaming App Logs

Page last updated:

This topic provides instructions for configuring some third-party log management services.

Once you have configured a service, refer to the Third-Party Log Management Services topic for instructions on binding your app to the service.

Logit.io

From your Logit.io dashboard:

1. Identify the Logit ELK stack you want to use.
2. Click Logstash Configuration.
3. Note your Logstash Endpoint.
4. Note your TCP-SSL, TCP, or UDP Port (not the syslog port).
5. Create the log drain service in Cloud Foundry.
   
   ```
   $ cf cups logit-ssl-drain -l syslog-tls://ENDPOINT:PORT
   
   or
   
   $ cf cups logit-drain -l syslog://ENDPOINT:PORT
   ```
6. Bind the service to an app.
   
   ```
   $ cf bind-service YOUR-CF-APP-NAME logit-ssl-drain
   
   or
   
   $ cf bind-service YOUR-CF-APP-NAME logit-drain
   ```
7. Restage or push the app using one of the following commands:
   
   ```
   $ cf restage YOUR-CF-APP-NAME
   
   $ cf push YOUR-CF-APP-NAME
   ```

After a short delay, logs begin to appear in Kibana.

Papertrail

From your Papertrail account:

1. Click Add System.

   ![Dashboard](Dashboard.png)

   Let's aggregate some logs. Add your first system in about 45 seconds, or take a tour.

2. Click the Other link.
3. Select I use Cloud Foundry, enter a name, and click Save.

4. Record the URL with port that is displayed after creating the system.

5. Create the log drain service in Cloud Foundry.

   ```bash
   cf cups my-logs -t syslog-disk://logs.papertrailapp.com:PORT
   ```

6. Bind the service to an app.

   ```bash
   cf bind-service APPLICATION-NAME my-logs
   ```

7. Restage the app.

   ```bash
   cf restage APPLICATION-NAME
   ```

   After a short delay, logs begin to flow automatically.

8. Once Papertrail starts receiving log entries, the view automatically updates to the logs viewing page.
Splunk

See [Streaming App Logs to Splunk](#) for details.

Splunk Storm

From your Splunk Storm account:

1. Click Add project.

2. Enter the project details.

3. Create a new input for Network data.
4. Manually enter the external IP addresses your Cloud Foundry administrator assigns to outbound traffic.

![Add network data](image1)

5. Note the host and port provided for TCP input.

![Authorized network inputs](image2)

6. Create the log drain service in Cloud Foundry using the displayed TCP host and port.

```
cf cups my-logs -l syslog://HOST:PORT
```

7. Bind the service to an app

```
cf bind-service APPLICATION-NAME my-logs
```

8. Restage the app

```
cf restage APPLICATION-NAME
```

After a short delay, logs begin to flow automatically.

9. Wait for some events to appear, then click Data Summary.

![Data Summary](image3)

10. Click the loggregator link to view all incoming log entries from Cloud Foundry.

```
```

![Data Summary](image4)

SumoLogic

Note: SumoLogic uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and later.

![SumoLogic](image5)

From your SumoLogic account:

1. Click the Add Collector link.
2. Choose *Hosted Collector* and fill in the details.

3. In the new collector’s row of the collectors view, click the *Add Source* link.

4. Select *HTTP* source and fill in the details. Note that you’ll be provided an HTTPS url

5. Once the source is created, a URL should be displayed. You can also view the URL by clicking the *Show URL* link beside the created source.
6. Create the log drain service in Cloud Foundry using the displayed URL.

   $ cf cups my-logs -l HTTPS-SOURCE-URL

7. Bind the service to an app.

   $ cf bind-service APPLICATION-NAME my-logs

8. Restage the app.

   $ cf restage APPLICATION-NAME

   After a short delay, logs begin to flow automatically.

9. In the SumoLogic dashboard, click Manage, then click Status to see a view of log messages received over time.

   ![SumoLogic dashboard](image)

10. In the SumoLogic dashboard, click Search. Place the cursor in the search box, then press Enter to submit an empty search query.

   ![SumoLogic search](image)

11. Logsene

   **Note:** Logsene uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and later.

   From your Sematext account:

   1. Click the [Create App / Logsene App](#) menu item. Enter a name and click Add Application to create the Logsene App.

   2. Create the log drain service in Cloud Foundry using the displayed URL.

   $ cf cups logsene-log-drain -l https://logsene-cf-receiver.sematext.com/YOUR_LOGSENE_TOKEN
3. Bind the log drain to an app. You could optionally bind multiple apps to one log drain.

```
$ cf bind-service YOUR-CF-APP-NAME logsene-log-drain
```

4. Restage the app.

```
$ cf restage APPLICATION-NAME
```

After a short delay, logs begin to flow automatically and appear in the Logsene UI.

---

Logentries is Not Supported

Cloud Foundry distributes log messages over multiple servers to handle load. Currently, we do not recommend using Logentries as it does not support multiple syslog sources.
Streaming App Logs to Splunk

Page last updated:

To integrate Cloud Foundry with Splunk Enterprise, complete the following process.

1. Create a Cloud Foundry Syslog Drain for Splunk

In Cloud Foundry, create a syslog drain user-provided service instance as described in Using Third-Party Log Management Services.

Choose one or more apps whose logs you want to drain to Splunk through the service.

Bind each app to the service instance and restart the app.

Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service. Locate the port number in the syslog URL. For example:

syslog://logs.example.com:1234

2. Prepare Splunk for Cloud Foundry

For detailed information about the following tasks, see the Splunk documentation.

Install the RFC5424 Syslog Technology Add-On

The Cloud Foundry Loggregator component formats logs according to the Syslog Protocol defined in RFC 5424. Splunk does not parse log fields according to this protocol. To allow Splunk to correctly parse RFC 5424 log fields, install the Splunk RFC5424 Syslog Technical Add-On.

Patch the RFC5424 Syslog Technology Add-On

1. SSH into the Splunk VM

2. Replace the old transforms.conf file with a new file that consists of the following text:

   ```
   [rfc5424_host]
   DEST_KEY = MetaData:Host
   REGEX = <d:1|a:1|S+r:1|S+/> FORMAT = host:$1
   
   [rfc5424_header]
   REGEX = <d:1|a:1|S+r:1|S+/>a:1|S+/>a:1|S+/>a:1|S+/>a:1|S+/> FORMAT = prival:$1 appname:$2 procid:$3 msgid:$4 MV_ADD = true
   ```

3. Restart Splunk

Create a TCP Syslog Data Input

Create a TCP Syslog Data Input in Splunk, with the following settings:

- **TCP port** is the port number you assigned to your log drain service
- **Set sourcetype** is Manual
- **Source type** is rfc5424_syslog (type this value into text field)
- **Index** is the index you created for your log drain service

Your Cloud Foundry syslog drain service is now integrated with Splunk.
3. Verify that Integration was Successful

Use Splunk to execute a query of the form:

```
sourcetype=rfc5424_syslog index=-THE-INDEX-YOU-CREATED appname=APP-GUID
```

To view logs from all apps at once, you can omit the `appname` field.

Verify that results rows contain the three Cloud Foundry-specific fields:

- **appname**: The GUID for the Cloud Foundry app
- **host**: The IP address of the Loggregator host
- **procid**: The Cloud Foundry component emitting the log

If the Cloud Foundry-specific fields appear in the log search results, integration is successful.

If logs from an app are missing, make sure that the following are true:

- The app is bound to the service and was restarted after binding
- The service port number matches the TCP port number in Splunk
Streaming App Logs with Fluentd

Fluentd is an open source collector that allows you to implement unified logging layers. With Fluentd, you can stream app logs to different backends or services like Elasticsearch, HDFS and Amazon S3. This topic explains how to integrate Fluentd with Cloud Foundry apps.

Step 1: Create a Cloud Foundry Syslog Drain for Fluentd

1. In Cloud Foundry, create a syslog drain user-provided service instance as described in Using Third-Party Log Management Services.
2. Choose one or more apps whose logs you want to drain to Fluentd through the service.
3. Bind each app to the service instance, and restart the app.
4. Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service.
5. Locate the port number in the syslog URL. For example:
   `syslog://logs.example.com:5140`

Step 2: Set up Fluentd for Cloud Foundry

This section assumes you have an active Fluentd instance running. If you do not have an active Fluentd instance, refer to the Fluentd Documentation/Install steps for more details.

Fluentd comes with native support for syslog protocol. To set up Fluentd for Cloud Foundry, configure the syslog input of Fluentd as follows.

1. In your main Fluentd configuration file, add the following `source` entry:
   ```xml
   <source>
     @type syslog
     port 5140
     bind 0.0.0.0
     tag cf.app
     protocol_type udp
   </source>
   ```
2. Restart the Fluentd service.

   **Note:** The Fluentd syslog input plugin supports `udp` and `tcp` options. Make sure to use the same transport that Cloud Foundry is using.

Fluentd will start listening for Syslog message on port 5140 and tagging the messages with `cf.app`, which can be used later for data routing. For more details about the full setup for the service, refer to the Config File article.

If your goal is to use an Elasticsearch or Amazon S3 backend, read the following guide: [http://www.fluentd.org/guides/recipes/elasticsearch-and-s3](http://www.fluentd.org/guides/recipes/elasticsearch-and-s3)
Streaming App Logs to Azure OMS Log Analytics (Beta)

This topic explains how to integrate your Cloud Foundry (CF) apps with OMS Log Analytics.

Operations Management Suite (OMS) Log Analytics is a monitoring service for Microsoft Azure. The OMS Log Analytics Firehose Nozzle is a CF component that forwards metrics from the Loggregator Firehose to OMS Log Analytics.

This topic assumes you are using the latest version of the Cloud Foundry Command Line Interface (cf CLI) and a working Pivotal Application Service (PAS) deployment on Azure.

Step 1: Create an OMS Workspace in Azure

See Get started with Log Analytics in the Microsoft Azure documentation to create an OMS workspace.

Step 2: Deploy the Nozzle to Cloud Foundry

1. Run `cf login -a https://api.YOUR-DOMAIN -u YOUR-USERNAME --skip-ssl-validation`, replacing `YOUR-DOMAIN` with your domain and `YOUR-USERNAME` with your CF username, to authenticate to your CF instance. For example:

   ```
   $ cf login -a https://api.example.com -u admin --skip-ssl-validation
   ```

2. Follow the steps below to create a new Cloud Foundry user and grant it access to the Loggregator Firehose using the UAA CLI (UAAC). For more information, see Creating and Managing Users with the UAA CLI (UAAC) and Orgs, Spaces, Roles, and Permissions.

   a. Use `uaac target uaa.YOUR-DOMAIN` to target your UAA server:

   ```
   $ uaac target uaa.example.com --skip-ssl-validation
   ```

   b. Run the following command to obtain an access token for the admin client:

   ```
   $ uaac token client get admin
   ```

   c. Run `cf create-user USERNAME PASSWORD`, replacing `USERNAME` with a new username and `PASSWORD` with a password, to create a new user. For example:

   ```
   $ cf create-user firehose-user firehose-password
   ```

   d. Run `uaac member add cloud_controller.admin USERNAME`, replacing `USERNAME` with the new username, to grant the new user admin permissions. For example:

   ```
   $ uaac member add cloud_controller.admin firehose-user
   ```

   e. Run `uaac member add doppler.firehose USERNAME`, replacing `USERNAME` with the new username, to grant the new user permission to read logs from the Loggregator Firehose endpoint. For example:

   ```
   $ uaac member add doppler.firehose firehose-user
   ```

3. Download the OMS Log Analytics Firehose Nozzle BOSH release from Github. Clone the repository and navigate to the `oms-log-analytics-firehose-nozzle` directory:

   ```
   $ git clone https://github.com/Azure/oms-log-analytics-firehose-nozzle.git
   $ cd oms-log-analytics-firehose-nozzle
   ```

4. Set the following environment variables in the OMS Log Analytics Firehose Nozzle manifest:

   ```
   warning: The OMS Log Analytics Firehose Nozzle is currently intended for evaluation and test purposes only. Do not use this product in a production environment.
   ```
<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applications:</td>
<td></td>
</tr>
<tr>
<td>- name: oms_nozzle</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>env:</td>
<td></td>
</tr>
<tr>
<td>OMS_WORKSPACE: YOUR-WORKSPACE-ID</td>
<td>Enter the ID and key value for your OMS workspace.</td>
</tr>
<tr>
<td>OMS_KEY: YOUR-OMS-KEY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>OMS_POST_TIMEOUT: 10s</td>
<td>(Optional) Set the HTTP post timeout for sending events to OMS Log Analytics. The default value is 10 seconds.</td>
</tr>
<tr>
<td>OMS_BATCH_TIME: 10s</td>
<td>(Optional) Set the interval for posting a batch to OMS. The default value is 10 seconds. For more information, see the Configure Additional Logging section below.</td>
</tr>
<tr>
<td>OMS_MAX_MSG_NUM_PER_BATCH: 1000</td>
<td>(Optional) Set the maximum number of messages to include in an OMS batch. The default amount is 1000. For more information, see the Configure Additional Logging section below.</td>
</tr>
<tr>
<td>FIREHOSE_USER: YOUR-FIREHOSE-USER</td>
<td>Enter the username and password for the Firehose user you created in Step 2c.</td>
</tr>
<tr>
<td>FIREHOSE_USER_PASSWORD: YOUR-FIREHOSE-PASSWORD</td>
<td></td>
</tr>
<tr>
<td>API_ADDR: <a href="https://api.YOUR-DOMAIN">https://api.YOUR-DOMAIN</a></td>
<td>Enter the URL of your API endpoint.</td>
</tr>
<tr>
<td>DOPPLER_ADDR: wss://doppler.YOUR-DOMAIN:443</td>
<td>Enter the URL of your Loggregator traffic controller endpoint.</td>
</tr>
<tr>
<td>EVENT_FILTER: YOUR-LIST</td>
<td>(Optional) Enter the event types you want to filter out in a comma-separated list. The valid event types are <code>METRIC</code>, <code>LOG</code>, and <code>HTTP</code>.</td>
</tr>
<tr>
<td>IDLE_TIMEOUT: 60s</td>
<td>(Optional) Set the duration for the Firehose keepalive connection. The default time is 60 seconds.</td>
</tr>
<tr>
<td>SKIP_SSL_VALIDATION: TRUE-OR-FALSE</td>
<td>(Optional) Change this value to allow insecure connections to the UAA and the traffic controller. To block insecure connections to the UAA and traffic controller, set this value to <code>FALSE</code>.</td>
</tr>
<tr>
<td>LOG_LEVEL: INFO</td>
<td>(Optional) Change this value to increase or decrease the amount of logs. Valid log levels in increasing order include <code>[INFO]</code>, <code>[ERROR]</code>, and <code>[DEBUG]</code>. The default value is <code>[INFO]</code>. For more information, see the Configure Additional Logging section below.</td>
</tr>
<tr>
<td>LOG_EVENT_COUNT: TRUE-OR-FALSE</td>
<td>Set this value to <code>TRUE</code> to log the total count of events that the nozzle has sent, received, and lost. OMS logs this value as <code>CounterEvents</code>.</td>
</tr>
<tr>
<td>LOG_EVENT_COUNT_INTERVAL: 60s</td>
<td>(Optional) Set the time interval for logging the event count to OMS. The default interval is 60 seconds. For more information, see the Configure Additional Logging section below.</td>
</tr>
</tbody>
</table>

5. Push the app:

```
cf push
```
Import the OMS View

1. From the main OMS Overview page, navigate to View Designer.
2. Click Import.
3. Click Browse.
5. Save the view. The main OMS Overview page displays the Tile.
6. Click the Tile to view visualized metrics.

See the OMS Log Analytics View Designer documentation for more information.

Create Alert Rules

See Overview of alerts in Microsoft Azure for more information about OMS Log Analytics alerts.

Set Alert Queries

This section includes example queries that operators can set in the OMS Portal.

- The following query alerts the operator when the nozzle sends a slowConsumerAlert to OMS:
  ```
  Type=CF_ValueMetric_CL Name_s=slowConsumerAlert
  ```
- The following query alerts the operator when Loggregator sends an LGR to indicate problems with the logging process:
  ```
  Type=CF_LogMessage_CL SourceType_s=LGR MessageType_s=ERR
  ```
- The following query alerts the operator when the number of lost events reaches a certain threshold, specified in the OMS Portal:
  ```
  Type=CF_CounterEvent_CL Job_s=nozzle Name_s=eventsLost
  ```
- The following query alerts the operator when the nozzle receives the TruncatingBuffer.DroppedMessages CounterEvent:
  ```
  Type=CF_CounterEvent_CL Name_s="TruncatingBuffer.DroppedMessages"
  ```

(Optional) Step 4: Configure Additional Logging

OMS Log Analytics Firehose Nozzle forwards metrics from the Loggregator Firehose to OMS with minimal processing, but the nozzle can push additional metrics to OMS.

Log Sent, Received, and Lost Events

If you set the LOG_EVENT_COUNT environment variable to TRUE in the manifest, the nozzle periodically sends the count of sent, received, and lost events to OMS. The value you set for the LOG_EVENT_COUNT_INTERVAL determines how frequently the nozzle sends the count.

Note: The nozzle does not count CounterEvents themselves in the sent, received, or lost event count.

The nozzle sends the count as a CounterEvent with a CounterKey of one of the following:

---

Note: The OMS view of Cloud Foundry is not yet available in the OMS Solutions Gallery. You can add it manually to view your logs in OMS Portal.
### Log Slow Consumer Alerts

Loggregator sends the nozzle a `slowConsumerAlert` in the following situations:

- WebSocket sends the error code `ClosePolicyViolation (1008)`
- The nozzle receives a `CounterEvent` with the value `TruncatingBuffer.DroppedMessages`

In either case, the nozzle sends the `slowConsumerAlert` event to OMS as the following `ValueMetric`:

<table>
<thead>
<tr>
<th>ValueMetric</th>
<th>MetricKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>nozzle.alert.slowConsumerAlert</td>
<td>1</td>
</tr>
</tbody>
</table>

See the *Slow Nozzle Alerts* section of the [Loggregator Guide for Cloud Foundry Operators](#) for more information.

### (Optional) Step 5: Scale the Deployment

#### Scale the Nozzle

If the nozzle is unable to keep up with processing logs from the Firehose, Loggregator alerts the nozzle. When the nozzle receives the alert, it sends a `slowConsumerAlert` to OMS. If this happens, scaling up the nozzle minimizes data loss.

If an operator chooses to scale up their deployment, the Firehose evenly distributes events across all instances of the nozzle. See the *Scaling Nozzles* section of the [Loggregator Guide for Cloud Foundry Operators](#) for more information.

Operators can create an alert rule for the `slowConsumerAlert` message.

#### Scale Loggregator

Loggregator sends `LGR` log messages to indicate problems with the logging process. See the *Scaling Loggregator* section of the [Loggregator Guide for Cloud Foundry Operators](#) for more information.

Operators can create an alert rule for the `LGR` message.
Managing Apps with the cf CLI

The following list provides information about managing apps with the Cloud Foundry Command Line Interface (cf CLI):

- Running Tasks
- Scaling an App Using cf scale
- Using Application Health Checks
Running Tasks

Page last updated:

This topic describes how to run tasks in Cloud Foundry. A task is an app or script whose code is included as part of a deployed app, but runs independently in its own container.

About Tasks

In contrast to a long running process (LRP), tasks run for a finite amount of time, then stop. Tasks run in their own containers and are designed to use minimal resources. After a task runs, Cloud Foundry destroys the container running the task.

As a single-use object, a task can be checked for its state and for a success or failure message.

Note: Running a task consumes an application instance, and will be billed accordingly.

Use Cases for Tasks

Tasks are used to perform one-off jobs, which include the following:

- Migrating a database
- Sending an email
- Running a batch job
- Running a data processing script
- Processing images
- Optimizing a search index
- Uploading data
- Backing-up data
- Downloading content

How Tasks Are Run

Tasks are always executed asynchronously, meaning that they run independently from the parent app or other tasks that run on the same app.

The life-cycle of a task is as follows:

1. A user initiates a task in Cloud Foundry using one of the following mechanisms:
   - `cf run-task APPNAME "TASK"` command. See the Running Tasks section of this topic for more information.
   - Cloud Controller v3 API call. See the Tasks API reference page for more information.
   - Cloud Foundry Java Client. See the Cloud Foundry Java Client Library and Cloud Foundry Java Client topics for more information.

2. Cloud Foundry creates a container specifically for the task.

3. Cloud Foundry runs the task on the container using the value passed to the `cf run-task` command.

4. Cloud Foundry destroys the container.

The container also inherits environment variables, service bindings, and security groups bound to the app.

Note: You cannot SSH into the container running a task.

Task Logging and Execution History

Any data or messages the task outputs to STDOUT or STDERR is available on the app’s firehose logs. A syslog drain attached to the app receives the task
The task execution history is retained for one month.

### Manage Tasks

At the system level, a user with admin-level privileges can use the Cloud Controller v3 API to view all tasks that are running within an org or space. For more information, see the documentation for the Cloud Controller v3 API.

In addition, admins can set the default memory and disk usage quotas for tasks on a global level. Initially, tasks use the same memory and disk usage defaults as apps. However, the default memory and disk allocations for tasks can be defined separately from the default app memory and disk allocations.

The default memory and disk allocations are defined using the Default App Memory and Default Disk Quota per App fields. These fields are available in the Application Developer Controls configuration screen of Pivotal Application Service (PAS).

### Run a Task on an App

You can use the Cloud Foundry Command Line Interface (cf CLI) to run a task in the context of an app.

**Note:** To run tasks with the cf CLI, you must install cf CLI v6.23.0 or later. See the [Installing the Cloud Foundry Command Line Interface](#) topic for information about downloading, installing, and uninstalling the cf CLI.

**Note:** To run a task without starting the app, push the app with `cf push -i 0` and then run the task. You can run the app later by scaling up its instance count.

To run a task on an app, perform the following steps:

1. Push your app:
   ```
   $ cf push APP-NAME
   ```

2. Run your task on the deployed app:
   ```
   $ cf run-task APP-NAME "TASK" --name TASK-NAME
   ```

The following example runs a database migration as a task on the `my-app` app:

```
$ cf run-task my-app "bin/rails db:migrate" --name my-task
Creating task for app my-app in org jdoe-org / space development as jdoe@pivotal.io... OK
Task 1 has been submitted successfully for execution.
```

**Note:** To re-run a task, you must run it as a new task using the above command.

Use the `cf logs APP-NAME --recent` command to display the recent logs of the app and all its tasks.

The following example displays the logs of a successful task:
The following example displays the logs of a failed task:

```
$ cf logs my-app --recent
2017-01-03T15:58:06.57-0800 [APP/TASK/my-task/0] OUT Creating container
2017-01-03T15:58:08.45-0800 [APP/TASK/my-task/0] OUT Successfully created container
2017-01-03T15:58:13.34-0800 [APP/TASK/my-task/0] ERR bash: bin/rails: command not found
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0] OUT Exit status 127
```

If your task name is unique, you can `grep` the output of the `cf logs` command for the task name to view task-specific logs.

## List Tasks Running on an App

To list the tasks for a given app, run the `cf tasks APP-NAME` command. For example:

```
$ cf tasks my-app
Getting tasks for app my-app in org jdoe-org / space development as jdoe@pivotal.io...
OK

id   name       state     start time      command
2    339044ef  FAILED     Wed, 23 Nov 2016 21:52:52 UTC  echo foo; sleep 100; echo bar
1    8d0618cf  SUCCEEDED Wed, 23 Nov 2016 21:37:26 UTC  bin/rails db:migrate
```

Each task has one of the following states:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNNING</td>
<td>The task is currently in progress.</td>
</tr>
<tr>
<td>FAILED</td>
<td>The task did not complete. This state occurs when a task does not work correctly or a user cancels the task.</td>
</tr>
<tr>
<td>SUCCEEDED</td>
<td>The task completed successfully.</td>
</tr>
</tbody>
</table>

## Cancel a Task

After running a task, you may be able to cancel it before it finishes. To cancel a running task, use the `cf terminate-task APP-NAME TASK-ID` command. For example:

```
$ cf terminate-task my-app 2
Terminating task 2 of app my-app in org jdoe-org / space development as jdoe@pivotal.io...
OK
```
Scaling an App Using cf scale

Factors such as user load, or the number and nature of tasks performed by an app, can change the disk space and memory the app uses. For many apps, increasing the available disk space or memory can improve overall performance. Similarly, running additional instances of an app can allow the app to handle increases in user load and concurrent requests. These adjustments are called scaling an app.

Use `cf scale` to scale your app up or down to meet changes in traffic or demand.

Note: You can configure your app to scale automatically based on rules that you set. See the Scaling an Application Using Autoscaler and Using the App Autoscaler CLI topics for more information.

Scaling Horizontally

Horizontally scaling an app creates or destroys instances of your app.

Incoming requests to your app are automatically load balanced across all instances of your app, and each instance handles tasks in parallel with every other instance. Adding more instances allows your app to handle increased traffic and demand.

Use `cf scale APP -i INSTANCES` to horizontally scale your app. Cloud Foundry will increase or decrease the number of instances of your app to match `INSTANCES`.

```
$ cf scale myApp -i 5
```

Scaling Vertically

Vertically scaling an app changes the disk space limit or memory limit that Cloud Foundry app to all instances of the app.

Use `cf scale APP -k DISK` to change the disk space limit applied to all instances of your app. `DISK` must be an integer followed by either an M, for megabytes, or G, for gigabytes.

```
$ cf scale myApp -k 512M
```

Use `cf scale APP -m MEMORY` to change the memory limit applied to all instances of your app. `MEMORY` must be an integer followed by either an M, for megabytes, or G, for gigabytes.

```
$ cf scale myApp -m 1G
```
Using Application Health Checks

This topic describes how to configure health checks for your apps in Cloud Foundry.

Overview

An application health check is a monitoring process that continually checks the status of a running Cloud Foundry app.

Developers can configure a health check for an app using the Cloud Foundry Command Line Interface (cf CLI) or by specifying the `health-check-http-endpoint` and `health-check-type` fields in an app manifest.

To configure a health check using the cf CLI, follow the instructions in the Configure Health Checks section below. For more information about using an app manifest to configure a health check, see the `health-check-http-endpoint` and `health-check-type` sections of the Deploying with App Manifest topic.

Application health checks function as part of the app lifecycle managed by Diego architecture.

Configure Health Checks

To configure a health check while creating or updating an app, use the `cf push` command:

```
cf push YOUR-APP -u HEALTH-CHECK-TYPE -t HEALTH-CHECK-TIMEOUT
```

Replace the placeholders in the example command above as follows:

- **HEALTH-CHECK-TYPE**: Valid health check types are `port`, `process`, and `http`. See the Health Check Types section below for more information.
- **HEALTH-CHECK-TIMEOUT**: The timeout is the amount of time allowed to elapse between starting up an app and the first healthy response. See the Health Check Timeouts section for more information.

Note: The health check configuration you provide with `cf push` overrides any configuration in the app manifest.

To configure a health check for an existing app or to add a custom HTTP endpoint, use the `cf set-health-check` command:

```
cf set-health-check YOUR-APP HEALTH-CHECK-TYPE --endpoint CUSTOM-HTTP-ENDPOINT
```

Replace the placeholders in the example command above as follows:

- **HEALTH-CHECK-TYPE**: Valid health check types are `port`, `process`, and `http`. See the Health Check Types section below for more information.
- **CUSTOM-HTTP-ENDPOINT**: A `http` health check defaults to using `/` as its endpoint, but you can specify a custom endpoint. See the Health Check HTTP Endpoints section below for more information.

Note: You can change the health check configuration of a deployed app with `cf set-health-check`, but you must restart the app for the changes to take effect.

Note: You can also use the `v3-set-health-check` CLI command to change the individual health check invocation timeout for an app. This option also requires restarting the app. See the Cloud Foundry CLI Reference Guide for more information.

Understand Health Checks

Health Check Lifecycle

The following table describes how application health checks work in Cloud Foundry.
<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>App developer deploys an app to Cloud Foundry.</td>
</tr>
<tr>
<td>2</td>
<td>When deploying the app, the developer specifies a health check type for the app and, optionally, a timeout. If the developer does not specify a health check type, then the monitoring process defaults to a <code>port</code> health check.</td>
</tr>
<tr>
<td>3</td>
<td>Cloud Controller stages, starts, and runs the app.</td>
</tr>
<tr>
<td>4</td>
<td>Based on the type specified for the app, Cloud Controller configures a health check that runs periodically for each app instance.</td>
</tr>
<tr>
<td>5</td>
<td>When Diego starts an app instance, the application health check runs every 2 seconds until a response indicates that the app instance is healthy or until the health check timeout elapses. The 2-second health check interval is not configurable.</td>
</tr>
<tr>
<td>6</td>
<td>When an app instance becomes healthy, its route is advertised, if applicable. Subsequent health checks are run every 30 seconds once the app becomes healthy. The 30-second health check interval is not configurable.</td>
</tr>
<tr>
<td>7</td>
<td>If a previously healthy app instance fails a health check, Diego considers that particular instance to be unhealthy. As a result, Diego stops and deletes the app instance, then reschedules a new app instance. This stoppage and deletion of the app instance is reported back to the Cloud Controller as a crash event.</td>
</tr>
<tr>
<td>8</td>
<td>When an app instance crashes, Diego immediately attempts to restart the app instance several times. After three failed restarts, Cloud Foundry waits 30 seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, Cloud Foundry stops trying to restart the app instance.</td>
</tr>
</tbody>
</table>

### Health Check Types

The following table describes the types of health checks available for apps and recommended circumstances in which to use them:

<table>
<thead>
<tr>
<th>Health Check Type</th>
<th>Recommended Use Case</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http</code></td>
<td>The app can provide an HTTP 200 response.</td>
<td>The <code>http</code> health check performs a GET request to the configured HTTP endpoint on the app's default port. When the health check receives an HTTP 200 response, the app is declared healthy. We recommend using the <code>http</code> health check type whenever possible. A healthy HTTP response ensures that the web app is ready to serve HTTP requests. The configured endpoint must respond within 1 second to be considered healthy. <strong>WARNING:</strong> To prevent false negatives, use a dedicated endpoint for health check where response time and result do not depend on business logic.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>The app can receive TCP connections (including HTTP web apps).</td>
<td>A health check makes a TCP connection to the port or ports configured for the app. For apps with multiple ports, a health check monitors each port. If you do not specify a health check type for your app, then the monitoring process defaults to a <code>port</code> health check. The TCP connection must be established within 1 second to be considered healthy.</td>
</tr>
<tr>
<td><code>process</code></td>
<td>The app does not support TCP connections (for example, a worker).</td>
<td>For a <code>process</code> health check, Diego ensures that any process declared for the app stays running. If the process exits, Diego stops and deletes the app instance.</td>
</tr>
</tbody>
</table>

### Health Check Timeouts

The value configured for the health check timeout is the amount of time allowed to elapse between starting up an app and the first healthy response from the app. If the health check does not receive a healthy response within the configured timeout, then the app is declared unhealthy.

In Pivotal Cloud Foundry, the default timeout is 60 seconds and the maximum configurable timeout is 600 seconds. You can modify the timeout in the **Application Containers** pane of the Pivotal Application Service (PAS) tile.

### Health Check HTTP Endpoints

Only used by `http` type, the `--endpoint` flag of the `cf set-health-check` command specifies the path portion of a URI that must be served by the app and return `HTTP 200` when the app is healthy.

**Note:** This command will only check the health of the default port of the app.
Note: For HTTP apps, we recommend setting the health check type to [http](http) instead of a simple port check.
Cloud Foundry Environment Variables

Page last updated:

Environment variables are the means by which the Cloud Foundry (CF) runtime communicates with a deployed app about its environment. This page describes the environment variables that the runtime and buildpacks set for app.

For information about setting your own app-specific environment variables, see the Environment Variable section of the Deploying with App Manifests topic.

View Environment Variables

Install the Cloud Foundry Command Line Interface (cf CLI), and use the `cf env` command to view the Cloud Foundry environment variables for your app. The `cf env` command displays the following environment variables:

- The `VCAP_APPLICATION` and `VCAP_SERVICES` variables provided in the container environment
- The user-provided variables set using the `cf set-env` command

```
$ cf env my-app
Getting env variables for app my-app in org my-org / space my-space as admin...
OK
System-Provided:

<table>
<thead>
<tr>
<th>VCAP_APPLICATION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>application_id: &quot;fa05c1a9-08c1-4bda-bae1-139850dec7a3&quot;,</td>
</tr>
<tr>
<td>application_name: &quot;my-app&quot;,</td>
</tr>
<tr>
<td>application_url: [ &quot;my-app.192.0.2.34.xip.io&quot; ],</td>
</tr>
<tr>
<td>application_version: &quot;fb8fbcc6-8d58-479e-bcc7-3b4ce5a70ca&quot;,</td>
</tr>
<tr>
<td>cfapps: &quot;<a href="https://api.example.com">https://api.example.com</a>&quot;,</td>
</tr>
<tr>
<td>limits: {</td>
</tr>
<tr>
<td>&quot;disk&quot;: 1024,</td>
</tr>
<tr>
<td>&quot;fds&quot;: 16384,</td>
</tr>
<tr>
<td>&quot;mem&quot;: 256</td>
</tr>
<tr>
<td>},</td>
</tr>
<tr>
<td>name: &quot;my-app&quot;,</td>
</tr>
<tr>
<td>space_id: &quot;06450c72-4669-4dc6-8096-45f9777db68a&quot;,</td>
</tr>
<tr>
<td>space_name: &quot;my-space&quot;,</td>
</tr>
<tr>
<td>uris: [ &quot;my-app.192.0.2.34.xip.io&quot; ],</td>
</tr>
<tr>
<td>users: null,</td>
</tr>
<tr>
<td>version: &quot;fb8fbcc6-8d58-479e-bcc7-3b4ce5a70ca&quot;</td>
</tr>
</tbody>
</table>
| }

User-Provided:

MY_DRAIN: http://drain.example.com
MY_ENV_VARIABLE: 100
```

App-Specific System Variables

The subsections that follow describe the environment variables that Cloud Foundry makes available to your app container. Some of these variables are the same across instances of a single app, and some vary from instance to instance.

You can access environment variables programmatically, including variables defined by the buildpack. For more information, refer to the buildpack documentation for Java, Node.js, and Ruby.

<table>
<thead>
<tr>
<th>Env Var</th>
<th>Running</th>
<th>Staging</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF_INSTANCE_ADDR</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>CF_INSTANCE_GUID</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_INSTANCE_INDEX</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>CF_INSTANCE_INTERNAL_IP</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Present</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CF_INSTANCE_ADDR</td>
<td></td>
<td></td>
<td>The CF_INSTANCE_IP and CF_INSTANCE_PORT of the app instance in the format IP:PORT.</td>
</tr>
<tr>
<td>CF_INSTANCE_GUID</td>
<td></td>
<td></td>
<td>The UUID of the particular instance of the app.</td>
</tr>
<tr>
<td>CF_INSTANCE_INDEX</td>
<td></td>
<td></td>
<td>The index number of the app instance.</td>
</tr>
<tr>
<td>CF_INSTANCE_IP</td>
<td></td>
<td></td>
<td>The external IP address of the host running the app instance.</td>
</tr>
<tr>
<td>CF_INSTANCE_INTERNAL_IP</td>
<td></td>
<td></td>
<td>The internal IP address of the container running the app instance.</td>
</tr>
</tbody>
</table>
CF_INSTANCE_PORT

The external, or host-side, port corresponding to the internal, or container-side, port with value PORT. This value is generally different from the PORT of the app instance.

Example: CF_INSTANCE_PORT=61045

CF_INSTANCE_PORTS

The list of mappings between internal, or container-side, and external, or host-side, ports allocated to the instance’s container. Not all of the internal ports are necessarily available for the app to bind to, as some of them may be used by system-provided services that also run inside the container. These internal and external values may differ.

Example: CF_INSTANCE_PORTS=[{external:61045,internal:8080},{external:61046,internal:2222}]

DATABASE_URL

For apps bound to certain services that use a database, CF creates a DATABASE_URL environment variable based on the VCAP_SERVICES environment variable at runtime.

CF uses the structure of the VCAP_SERVICES environment variable to populate DATABASE_URL. CF recognizes any service containing a JSON object with the following form as a candidate for DATABASE_URL and uses the first candidate it finds.

```
{
  "some-service": [
    { "credentials": {
      "uri": "SOME-DATABASE-URL"
    }
  }
}
```

For example, consider the following VCAP_SERVICES:

```
VCAP_SERVICES =
{
  "elephantsql": [
    { "name": "elephantsql-c6c60",
      "label": "elephantsql",
      "credentials": {
        "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb"
      }
    }
  ]
}
```

Based on this VCAP_SERVICES, CF creates the following DATABASE_URL environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb
```

HOME

Root folder for the deployed app.

Example: HOME=/home/vcap/app

LANG

LANG is required by buildpacks to ensure consistent script load order.
MEMORY_LIMIT

The maximum amount of memory that each instance of the app can consume. You specify this value in an app manifest or with the cf CLI when pushing an app. The value is limited by space and org quotas.

If an instance exceeds the maximum limit, it will be restarted. If Cloud Foundry is asked to restart an instance too frequently, the instance will instead be terminated.

Example: `MEMORY_LIMIT=512M`

PORT

The port on which the app should listen for requests. The Cloud Foundry runtime allocates a port dynamically for each instance of the app, so code that obtains or uses the app port should refer to it using the `PORT` environment variable.

Example: `PORT=8080`

PWD

Identifies the present working directory, where the buildpack that processed the app ran.

Example: `PWD=/home/vcap/app`

TMPDIR

Directory location where temporary and staging files are stored.

Example: `TMPDIR=/home/vcap/tmp`

USER

The user account under which the app runs.

Example: `USER=vcap`

VCAP_APP_PORT

Deprecated name for the `PORT` variable defined above.

VCAP_APPLICATION

This variable contains the associated attributes for a deployed app. Results are returned in JSON format. The table below lists the attributes that are returned.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applicaton_id</td>
<td>GUID identifying the app.</td>
</tr>
<tr>
<td>applicaton_name</td>
<td>The name assigned to the app when it was pushed.</td>
</tr>
<tr>
<td>applicaton_uris</td>
<td>The URIs assigned to the app.</td>
</tr>
</tbody>
</table>
### Application Version

GUID identifying a version of the app. Each time an app is pushed or restarted, this value is updated.

### CF API Location

Location of the Cloud Controller API for the CF Deployment where the app runs.

### Host

Deprecated. IP address of the app instance.

### Limits

The limits to disk space, number of files, and memory permitted to the app. Memory and disk space limits are supplied when the app is deployed, either on the command line or in the app manifest. The number of files allowed is operator-defined.

### Name

Identical to `application_name`.

### Space ID

GUID identifying the app’s space.

### Space Name

Human-readable name of the space where the app is deployed.

### Start

Human-readable timestamp for the time the instance was started. Not provided on Diego Cells.

### Started at

Identical to `start`. Not provided on Diego Cells.

### Started at Timestamp

Unix epoch timestamp for the time the instance was started. Not provided on Diego Cells.

### State Timestamp

Identical to `started_at_timestamp`. Not provided on Diego Cells.

### URIs

Identical to `application_uris`. You must ensure that both `application_uris` and `uris` are set to the same value.

### Users

Deprecated. Not provided on Diego Cells.

### Version

Identical to `application_version`.

The following example shows how to set the `VCAP_APPLICATION` environment variable:

```json
VCAP_APPLICATION={"instance_id":"fe98dc76ba549876543210abcd1234", "instance_index":0,"host":"0.0.0.0","port":61857,"started_at":"2013-08-12 00:05:29 +0000","started_at_timestamp":1376265929,"start":"2013-08-12 00:05:29 +0000","state_timestamp":1376265929,"limits":{"mem":512,"disk":1024,"fds":16384},"application_version":"ab12cd34-5678-abcd-0123-abcdef987654","application_name":"styx-james","application_uris": ["my-app.example.com"],"version":"ab12cd34-5678-abcd-0123-abcdef987654","name":"my-app","uris": ["my-app.example.com"],"users":null}
```

### VCAP_SERVICES

For bindable services, Cloud Foundry adds connection details to the `VCAP_SERVICES` environment variable when you restart your app, after binding a service instance to your app.

The results are returned as a JSON document that contains an object for each service for which one or more instances are bound to the app. The service object contains a child object for each service instance of that service that is bound to the app. The attributes that describe a bound service are defined in the table below.

The key for each service in the JSON document is the same as the value of the “label” attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binding_name</td>
<td>The name assigned to the service binding by the user.</td>
</tr>
<tr>
<td>instance_name</td>
<td>The name assigned to the service instance by the user.</td>
</tr>
<tr>
<td>name</td>
<td>The <code>binding_name</code> if it exists; otherwise the <code>instance_name</code>.</td>
</tr>
<tr>
<td>label</td>
<td>The name of the service offering.</td>
</tr>
<tr>
<td>tags</td>
<td>An array of strings an app can use to identify a service instance.</td>
</tr>
<tr>
<td>plan</td>
<td>The service plan selected when the service instance was created.</td>
</tr>
<tr>
<td>credentials</td>
<td>A JSON object containing the service-specific credentials needed to access the service instance.</td>
</tr>
</tbody>
</table>

To see the value of VCAP_SERVICES for an app pushed to Cloud Foundry, see [View Environment Variable Values](#).

The example below shows the value of VCAP_SERVICES for bound instances of several services available in the Pivotal Web Services Marketplace.
Environment Variable Groups

Environment variable groups are system-wide variables that enable operators to apply a group of environment variables to all running apps and all staging apps separately.

An environment variable group consists of a single hash of name-value pairs that are later inserted into an app container at runtime or at staging. These values can contain information such as HTTP proxy information. The values for variables set in an environment variable group are case-sensitive.

When creating environment variable groups, consider the following:

- Only the Cloud Foundry operator can set the hash value for each group.
- All authenticated users can get the environment variables assigned to their app.
- All variable changes take effect after the operator restarts or restages the apps.
- Any user-defined variable takes precedence over environment variables provided by these groups.

The table below lists the commands for environment variable groups.

<table>
<thead>
<tr>
<th>CLI Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>running-environment-variable-group or rrevg</td>
<td>Retrieves the contents of the running environment variable group.</td>
</tr>
<tr>
<td>staging-environment-variable-group or sevg</td>
<td>Retrieves the contents of the staging environment variable group.</td>
</tr>
<tr>
<td>set-staging-environment-variable-group or ssevg</td>
<td>Passes parameters as JSON to create a staging environment variable group.</td>
</tr>
<tr>
<td>set-running-environment-variable-group or srevg</td>
<td>Passes parameters as JSON to create a running environment variable group.</td>
</tr>
</tbody>
</table>

The following examples demonstrate how to retrieve the environment variables:
$ cf revg
Retrieving the contents of the running environment variable group as sampledeveloper@example.com...
OK
Variable Name  Assigned Value
HTTP Proxy  198.51.100.130

$ cf sevg
Retrieving the contents of the staging environment variable group as sampledeveloper@example.com...
OK
Variable Name  Assigned Value
HTTP Proxy  203.0.113.105
EXAMPLE-GROUP  2001

$ cf apps
Getting apps in org SAMPLE-ORG-NAME / space dev as sampledeveloper@example.com...
OK
name  requested state  instances  memory  disk  urls
my-app  started  1/1  256M  1G  my-app.com

$ cf env APP-NAME
Getting env variables for app APP-NAME in org SAMPLE-ORG-NAME / space dev as sampledeveloper@example.com...
OK
System-Provided:

```
{
  "VCAP_APPLICATION": {
    "application_name": "APP-NAME",
    "application_uris": [
      "my-app.example.com"
    ],
    "application_version": "7d0d94d1-7b6f-406a-9d21-506463147ed3",
    "limits": {
      "disk": 1024,
      "dur": 16384,
      "mem": 256
    },
    "name": "APP-NAME",
    "space_id": "37189599-2407-9946-865e-8ebd0e2df89a",
    "space_name": "dev",
    "uris": [
      "my-app.example.com"
    ],
    "users": null,
    "version": "7d0d94d1-7b6f-406a-9d21-506463147ed3"
  }
}
```

Running Environment Variable Groups:
HTTP Proxy: 198.51.100.130

Staging Environment Variable Groups:
EXAMPLE-GROUP: 2001
HTTP Proxy: 203.0.113.105

The following examples demonstrate how to set environment variables:

$ cf sevg "{"test":"198.51.100.130","test2":"203.0.113.105"}"
Setting the contents of the staging environment variable group as admin...
OK

$ cf sevg
Retrieving the contents of the staging environment variable group as admin...
OK
Variable Name  Assigned Value
test  198.51.100.130
test2  203.0.113.105

$ cf revg "{"test3":"2001","test4":"2010"}"
Setting the contents of the running environment variable group as admin...
OK

$ cf revg
Retrieving the contents of the running environment variable group as admin...
OK
Variable Name  Assigned Value
test3  2001
test4  2010
Cloud Controller API Client Libraries

This topic describes the client libraries available for developers who want to consume the Cloud Controller API (CAPI).

Overview

CAPI is the entry point for most operations within the Cloud Foundry (CF) platform. You can use it to manage orgs, spaces, and apps, which includes user roles and permissions. You can also use CAPI to manage the services provided by your CF deployment, including provisioning, creating, and binding them to apps.

For more information, see the [CAPI documentation](#).

Client Libraries

While you can develop apps that consume CAPI by calling it directly as in the API documentation, you may want to use an existing client library. See the available client libraries below.

Supported

CF currently supports the following clients for CAPI:

- [Java](#)
- [Scripting](#) with the Cloud Foundry Command Line Interface (cf CLI)

Experimental

The following client is experimental and a work in progress:

- [Golang](#)

Unofficial

CF does not support the following clients, but they may be supported by third-parties:

- [Golang](#)
- [Golang](#)
- [Node.js](#)
- [Python](#):
  - [hsdp/python-cf-api](#)
  - [cloudfoundry-community/cf-python-client](#)
Considerations for Designing and Running an App in the Cloud

Page last updated:

App Design for the Cloud

Apps written in supported frameworks often run unmodified on Cloud Foundry if the app design follows a few simple guidelines. Following these guidelines facilitates app deployment to Cloud Foundry and other cloud platforms.

The following guidelines represent best practices for developing modern apps for cloud platforms. For more detailed reading about good app design for the cloud, see

The Twelve-Factor App.

For more information about the features of HTTP routing handled by the Cloud Foundry router, see the HTTP Routing topic. For more information about the lifecycle of app containers, see the App Container Lifecycle topic.

Avoid Writing to the Local File System

Apps running on Cloud Foundry should not write files to the local file system for the following reasons:

- **Local file system storage is short-lived.** When an app instance crashes or stops, the resources assigned to that instance are reclaimed by the platform including any local disk changes made since the app started. When the instance is restarted, the app will start with a new disk image. Although your app can write local files while it is running, the files will disappear after the app restarts.

- **Instances of the same app do not share a local file system.** Each app instance runs in its own isolated container. Thus a file written by one instance is not visible to other instances of the same app. If the files are temporary, this should not be a problem. However, if your app needs the data in the files to persist across app restarts, or the data needs to be shared across all running instances of the app, the local file system should not be used. We recommend using a shared data service like a database or blobstore for this purpose.

For example, instead of using the local file system, you can use a Cloud Foundry service such as the MongoDB document database or a relational database like MySQL or Postgres. Another option is to use cloud storage providers such as Amazon S3, Google Cloud Storage, Dropbox, or Box. If your app needs to communicate across different instances of itself, consider a cache like Redis or a messaging-based architecture with RabbitMQ.

If you must use a file system for your app because, for example, your app interacts with other apps through a network attached file system, or because your app is based on legacy code that you cannot rewrite, consider using Volume Services to bind a network attached file system to your app.

Cookies Accessible across Apps

In an environment with shared domains, cookies might be accessible across apps.

Many tracking tools such as Google Analytics and Mixpanel use the highest available domain to set their cookies. For an application using a shared domain such as example.com, a cookie set to use the highest domain has a Domain attribute of .example.com in its HTTP response header. For example, an app at my-app.shared-domain.example.com might be able to access the cookies for an app at your-app.shared-domain.example.com.

You should decide whether or not you want your apps or tools that use cookies to set and store the cookies at the highest available domain.

Port Considerations

Clients connect to apps running on Cloud Foundry by making requests to URLs associated with the app. Cloud Foundry allows HTTP requests to apps on ports 80 and 443. For more information, see the Routes and Domains topic.

Cloud Foundry also supports WebSocket handshake requests over HTTP containing the Upgrade header. The Cloud Foundry router handles the upgrade and initiates a TCP connection to the app to form a WebSocket connection.

To support WebSockets, the operator must configure the load balancer correctly. Depending on the configuration, clients may have to use a different port for WebSocket connections, such as port 4443, or a different domain name. For more information, see the Supporting WebSockets topic.

Cloud Foundry Updates and Your App
For app management purposes, Cloud Foundry may need to stop and restart your app instances. If this occurs, Cloud Foundry performs the following steps:

1. Cloud Foundry sends a single termination signal to the root process that your start command invokes.
2. Cloud Foundry waits 10 seconds to allow your app to cleanly shut down any child processes and handle any open connections.
3. After 10 seconds, Cloud Foundry forcibly shuts down your app.

Your app should accept and handle the termination signal to ensure that it shuts down gracefully. To achieve this, the app is expected to follow the steps below when shutting down:

1. App receives termination signal
2. App closes listener so that it stops accepting new connections
3. App finishes serving in-flight requests
4. App closes existing connections as their requests complete
5. App shuts down or is killed

See the Sample HTTP App GitHub repository for an implementation of the expected shutdown behavior in Golang.

Ignore Unnecessary Files When Pushing

By default, when you push an app, all files in the app’s project directory tree are uploaded to your Cloud Foundry instance, except version control and configuration files or folders with the following names:

- .cfignore
- _.darcs
- .DS_Store
- .git
- .gitignore
- .hg
- manifest.yml
- .svn

In addition to these, if API request diagnostics are directed to a log file and the file is within the project directory tree, it is excluded from the upload. You can direct these API request diagnostics to a log file using `cf config --trace` or the `CF_TRACE` environment variable.

If the app directory contains other files, such as `temp` or `log` files, or complete subdirectories that are not required to build and run your app, you might want to add them to a `.cfignore` file to exclude them from upload. Especially with a large app, uploading unnecessary files can slow app deployment.

To use a `.cfignore` file, create a text file named `.cfignore` in the root of your app directory structure. In this file, specify the files or file types you wish to exclude from upload. For example, these lines in a `.cfignore` file exclude the “tmp” and “log” directories.

```plaintext
tmp
log
```

The file types you will want to exclude vary, based on the app frameworks you use. For examples of commonly-used `.gitignore` files, see https://github.com/github/gitignore.

Run Multiple Instances to Increase Availability

Singleton apps may become temporarily unavailable for reasons that include:

- During an upgrade, Pivotal Cloud Foundry (PCF) gracefully shuts down the apps running on each Diego cell and then restarts them on another Diego cell. Single app instances may become temporarily unavailable if the replacement instance does not become healthy within the cell’s evacuation timeout, which defaults to 10 minutes.
Unexpected faults in PCF system components or underlying infrastructure, such as container-host VMs or IaaS Availability Zones, may cause lone app instances to disappear or become unroutable for a minute or two.

To avoid the risk of an app becoming temporarily unavailable, developers can run more than one instance of the app.

Using Buildpacks

A buildpack consists of bundles of detection and configuration scripts that provide framework and runtime support for your apps. When you deploy an app that needs a buildpack, Cloud Foundry installs the buildpack on the Diego cell where the app runs.

For more information, see the Buildpacks topic.
PCF Security and Compliance Guide

For Security Professionals and PCF Users

This guide explains how Pivotal Cloud Foundry (PCF) manages network access, roles and permissions, internal communications, container hardening, and other security issues. It is intended to give security professionals a complete view of PCF security, and to help all PCF users, not just the security experts, keep the platform secure.

In addition, this guide provides information about PCF compliance with published control standards and regulations such as NIST Special Publication 800-53(r4) and GDPR.

Pivotal Security Processes and CVE Reports

Pivotal publishes security updates regularly in response to privately- and publicly-reported Common Vulnerabilities and Exposures (CVEs).

- See the latest CVEs on the Pivotal Application Security Team page.
- To learn about Pivotal’s vulnerability reporting and responsible disclosure process, read PCF Security Overview and Policy.
- To learn about the testing, release and security lifecycle of PCF, see PCF Testing, Release, and Security Lifecycle.

Security

- Security Concepts: Provides links to conceptual documentation about how security is implemented in PCF.
- PCF Infrastructure Security: Provides guidance and procedures for securing PCF infrastructure such as hardening stemcells and managing the certificates that enable TLS communication.
- Network Security: Covers the security aspects of PCF networking such as the paths, ports, and protocols that components use to communicate.
- Credential and Identity Management: Describes how PCF manages permissions and trust for PCF user accounts. Also provides documentation about CredHub, the credential management system that BOSH uses to store deployment credentials and that PCF runtimes use to create and manage app and service credentials.
- Security for Apps and Services: Collects documentation about the security mechanisms that surround apps and services running on PCF.
- Certificates on PCF: How certificates are utilized in PCF to secure both internal and external networking calls.
- Security Processes and Stemcells: How Pivotal responds to security vulnerabilities, and how it tests and updates the versioned operating systems that its products run on.

Compliance

- NIST Controls and PCF: Provides a dedicated site that assesses Pivotal Cloud Foundry against NIST SP 800-53(r4) Controls.
- General Data Protection Regulation: Provides an overview of the General Data Protection Regulation (GDPR) and where Pivotal Cloud Foundry (PCF) may store personal data.
Security Processes

This section explains how Pivotal responds to security vulnerabilities, and how it tests and updates its stemcells, the versioned operating systems that its products run on.

- **Pivotal Cloud Foundry Security Overview and Policy**: Covers Pivotal’s responsible disclosure and vulnerability response procedures for the Pivotal Cloud Foundry (PCF) platform.
- **PCF Testing, Release, and Security Lifecycle**: Explains how Pivotal’s practices, tools, and organizational structures work together to create and support stable releases of PCF.
- **PKS Security Disclosure and Release Process**: Describes the processes for disclosing security issues and releasing related fixes for Pivotal Container Service (PKS), Kubernetes, Cloud Foundry Container Runtime (CFCR), VMware NSX, and VMware Harbor.
Pivotal Cloud Foundry Security Overview and Policy

This document outlines our security policy and is addressed to operators deploying Pivotal Cloud Foundry (PCF) using Pivotal Cloud Foundry Operations Manager.

For a comprehensive overview of the security architecture of each PCF component, refer to the Cloud Foundry Security topic.

How Pivotal Monitors for Security Vulnerabilities

Pivotal receives private reports on vulnerabilities from customers and from field personnel via our secure disclosure process. We also monitor public repositories of software security vulnerabilities to identify newly discovered vulnerabilities that might affect one or more of our products.

How to Report a Vulnerability

Pivotal encourages users who become aware of a security vulnerability in our products to contact Pivotal with details of the vulnerability. Please send descriptions of any vulnerabilities found to security@pivotal.io. Please include details on the software and hardware configuration of your system so that we can reproduce the issue.

Note: We encourage use of encrypted email. Our public PGP key is located at http://www.pivotal.io/security.

Notification Policy

PCF has many customer stakeholders who need to know about security updates. When there is a possible security vulnerability identified for a PCF component, we do the following:

1. Assess the impact to PCF.
2. If the vulnerability would affect a PCF component, we schedule an update for the impacted component(s).
3. Update the affected component(s) and perform system tests.
4. Announce the fix publicly via the following channels:
   a. Automated notification to end users who have downloaded or subscribed to a PCF product on Pivotal Network when a new, fixed version is available.

Classes of Vulnerabilities

Attackers can exploit vulnerabilities to compromise user data and processing resources. This can affect data confidentiality, integrity, and availability to different degrees. For vulnerabilities related to Ubuntu provided packages, Pivotal follows Canonical’s priority levels. For other vulnerabilities, Pivotal follows Common Vulnerability Scoring System v3.0 standards when assessing severity.

Pivotal uses Canonical’s Ubuntu distribution of Linux for PCF Ubuntu stemcells and roots. Canonical provides Pivotal with support services allowing us to escalate CVEs that we determine may affect PCF. In general, Pivotal does not escalate to upstream open source software components or vendors for Medium or Low CVEs that are not yet patched. PCF may escalate on behalf of a customer for High or Critical CVEs. PCF customers who are interested in addressing CVEs in Ubuntu that are not yet patched can establish their own support relationship with Canonical.

Pivotal reports the severity of vulnerabilities using the following severity classes:

High

High severity vulnerabilities are those that can be exploited by an unauthenticated or authenticated attacker, from the Internet or those that break the guest/host Operating System isolation. The exploitation could result in the complete compromise of confidentiality, integrity, and availability of user data...
and/or processing resources without user interaction. Exploitation could be leveraged to propagate an Internet worm or execute arbitrary code between Virtual Machines and/or the Host Operating System. This rating also applies to those vulnerabilities that could lead to the complete compromise of availability when the exploitation is by a remote unauthenticated attacker from the Internet or through a breach of virtual machine isolation.

Moderate

Moderate vulnerabilities are those in which the ability to exploit is mitigated to a significant degree by configuration or difficulty of exploitation, but in certain deployment scenarios could still lead to the compromise of confidentiality, integrity, or availability of user data and/or processing resources.

Low

Low vulnerabilities are all other issues that have a security impact. These include vulnerabilities for which exploitation is believed to be extremely difficult, or for which successful exploitation would have minimal impact.

Release Policy

PCF schedules regular monthly releases of software in the PCF Suite to address Low / Medium severity vulnerability exploits. When High severity vulnerability exploits are identified, PCF releases fixes to software in the PCF Suite on-demand, with as fast a turnaround as possible.

Alerts/Actions Archive

http://www.pivotal.io/security
PCF Testing, Release, and Security Lifecycle

This topic explains how Pivotal’s development practices, automated build tools, and organizational structures work together to create and support stable releases of Pivotal Cloud Foundry (PCF).

Summary

- PCF teams building system components receive frequent feedback, which helps to secure code from exposure to vulnerability.
- Every PCF release follows a strict workflow and passes through numerous quality and compliance checks before distribution.
- Teams build tests into the product consistently and run them automatically with any code change.

Release Mechanics

Pivotal releases, patches, and supports multiple versions of PCF simultaneously. This section explains the versioning and support conventions Pivotal follows.

Versioning

Pivotal numbers PCF releases following a semantic versioning style format, X.Y.Z, where X and Y indicate major and minor releases, and Z designates patch releases. Major and minor releases change PCF functionality; patch releases are backward-compatible security patches or bug fixes.

Support

As of PCF 1.8, Pivotal supports each major and minor PCF release according to the following criteria:

- Pivotal supports the release for at least 9 months following its first publication date.
- Pivotal supports the last three major or minor releases, even if this extends coverage beyond 9 months.

Support includes maintenance updates and upgrades, bug and security fixes, and technical assistance. The Pivotal Support Policy describes support standards, technical guidance, and publication phases in more detail. The Pivotal Support Services Terms and Conditions defines Pivotal support in legal terms.

Patch Releases

Patch releases are more frequent and less predictable than major/minor releases. The v1.6.x line provides a good example of their frequency. PCF 1.6.1 was released on October 26, 2015. Through August 2016, 36 additional patches of Elastic Runtime 1.6.x and 18 patches of Ops Manager 1.6.x provided security and bug fixes to customers.

Pivotal identifies security issues using standard nomenclature from Common Vulnerabilities and Exposures (CVE), Ubuntu Security Notices (USN), and other third-party sources. Read about security fixes in core Cloud Foundry code or packaged dependencies in the release notes for Ops Manager and Pivotal Application Service (PAS).

Pivotal.io/security maintains a running list of security fixes in PCF and PCF dependencies. Consult that page to see the most recent findings from Pivotal’s security team.

Upgrading

All PCF releases pass through extensive test suites that include automated unit, integration, and acceptance tests on multiple IaaSes. Regardless of this extensive testing, Pivotal recommends that you test major and minor releases in a non-production environment before implementing them across your deployment. Upgrade your production environment as soon as possible after you validate the new release on your test environment.
Release Testing, Integration, and Validation

This section describes Pivotal's software development processes and explains compliance and regulatory standards to which Pivotal software adheres.

Test-Driven Development

Pivotal's development process relies on a strict workflow with continuous automated testing. Pivotal R&D does not separate engineering and testing teams. Rather, every Pivot on each engineering team is responsible for ensuring the quality of their code. They write tests for all of the software components that they develop, often before writing the software itself.

With every software change, automated build pipelines trigger these tests for the new software component and for everything it touches. If a new code check-in does not pass its tests or causes a failure elsewhere, it pauses the build pipeline for the entire team, or sometimes all of Pivotal R&D. The transparency of this process encourages developers to work together to address code issues quickly.

Pivotal applies the following automated testing approaches, scenarios, and frameworks to PCF components and to the release as a whole:

- **Unit tests**: Development teams write unit tests to express and validate desired functional behavior of product components. Typical frameworks used are RSpec and Ginkgo. These tests run continuously throughout the development cycle.

- **OSS integration tests**: The Release Integration team exercises a full deployment of open-source Cloud Foundry to validate all end-user features. They maintain the Cloud Foundry Acceptance Test (CATs) suite alongside the OSS cf-release. Cloud Foundry component teams also contribute acceptance test suites at the OSS Integration Test level. These tests exercise and validate their components' functional, performance, and integration health.

- **PCF integration tests**: The PCF Release Engineering (RelEng) team validates the quality and cross-product integration health of the commercial PCF release. RelEng runs OSS Acceptance Tests against all supported releases. These tests run on full PCF instances configured to represent diverse real-world customer scenarios on various IaaSes and using both internal and external load balancer, database, blobstore, and user store solutions.

Additional Pre-Release Gates: Internal, PWS, and Compliance

In addition to its automated unit and integration testing, Pivotal deploys all upgrades slated for upcoming PCF releases on at-scale test environments. Prior to each Major or Minor commercial release, Pivotal runs the entire Pivotal Cloud Foundry Suite of services on several internally-managed large integration environments that run customer-like data and workloads.

Pivotal also pushes upcoming PCF feature upgrades and patches to its Pivotal Web Services platform, where customers continually deploy and host hundreds of mission-critical applications at scale, 24/7. The PWS environment gives Pivotal a continuous source of real-world usage and performance metrics that inform product development teams.

All PCF product teams participate in go-to-market steps for each release, as is often required for shipping a legally compliant product. Examples include Open Source License File attribution and an Export Compliance classification.

Patch Releases: Security and Bug Fixes

Pivotal uses established processes to track, disclose, and remediate vulnerabilities in PCF and related dependent components. This section explains how Pivotal identifies vulnerabilities and implements fixes for them.

Identifying Security Vulnerabilities

Pivotal has an established process to track and patch vulnerabilities in software dependencies and PCF software. Additionally, pivotal.io/security describes a responsible disclosure process for reporting vulnerabilities identified in Pivotal software by 3rd parties.

Pivotal uses multiple methods to identify security vulnerabilities in Pivotal software and dependencies internally, including:

- Security notifications from Canonical for their Ubuntu operating system, provided through Pivotal's commercial relationship with Canonical
- Software component scans several times per day, using 3rd party security software which updates continuously from external security vulnerability sources
- Dependency analysis software that identifies and catalogs software dependencies
- Security vulnerability notifications from known software dependencies

Pivotal also monitors externally-reported vulnerabilities from many sources, including:
- 3rd party security analysis requested by Pivotal
- Cloud Foundry Foundation security notifications from member companies
- Customer, prospect and other 3rd party security reports

When Pivotal discovers a potential security vulnerability in PCF, the security team opens an issue to assess it. If it confirms the vulnerability exists, Pivotal identifies and updates affected components with plans to backport the fix to stable releases. Fixes are implemented on a target timeline based on the severity level of the vulnerability.

### Fix, Test, and Release Lifecycle

This flowchart details the steps that Pivotal performs on a typical high-priority CVE, to publish a patch release fix on https://network.pivotal.io.
Security Concepts

This section provides links to conceptual documentation that describes how Pivotal Cloud Foundry (PCF) implements security at different levels of the platform.

- **Cloud Foundry Security**: Provides an overview of the measures Cloud Foundry implements to minimize security risks.
- **Container Security**: Describes how PCF isolates containers and limits privileges for containers.
- **Container-to-Container Networking**: Illustrates how the Container-to-Container Networking feature enables and secures internal app-to-app communication.
- **Application Security Groups**: Explains the different types of application security groups, how they work and how to apply them.
- **Application SSH Components and Processes**: Provides an overview of the components that support SSH access to apps.
- **Floating Stemcells**: Describes how PCF automatically upgrades all compatible products when a new stemcell is available.
PCF Infrastructure Security

This section provides links to topics about security infrastructure in Pivotal Cloud Foundry (PCF).

- **Stemcell Security**: Details how Pivotal tests and updates its stemcells, which supply the base versioned operating system for all PCF products.
- **Certificates and TLS in PCF**: Provides information on how to manage the certificates that enable TLS communications in PCF.
- **Disk Encryption**: Describes how to encrypt disks or rotate disk encryption keys in a PCF deployment.
- **Security Event Logging**: Provides a reference of security-related events logged by the Cloud Controller, CredHub and UAA components in PCF.
Stemcell Security

This section provides links to topics about how Pivotal implements security in the stemcell layer of Pivotal Cloud Foundry (PCF).

- **Floating Stemcells**: How PCF automatically upgrades all compatible products when a new stemcell is available.
- **Windows Stemcell Hardening**: The settings for Local Group Policy and Local Security Policy that Pivotal incorporates into its Windows 2016 stemcells to optimize security.
- **Linux Stemcell Hardening**: How Pivotal secures Linux stemcells through regular testing and minimizing their surface of vulnerability.

For more information about stemcell availability and related security fixes, see [Stemcell Release Notes](#).
Linux Stemcell Hardening

Note: This document applies to stemcell v3263 and later.

Customers and prospects often ask for details on stemcell hardening, i.e., the process by which we secure Pivotal Cloud Foundry by reducing its vulnerability surface from outside access. This document provides responses to some commonly-asked questions regarding the security configuration enhancements and hardening tests that Pivotal applies to the Cloud Foundry (“CF”) stemcell. This information will be helpful to customer accreditation teams who are responsible for running configuration scans of a Cloud Foundry deployment, and also to auditors who need a documentation artifact to feed into the customers’ existing security assessment processes.

1. **WHAT IS A STEMCELL?** A stemcell is a versioned Operating System (“OS”) image wrapped with IaaS specific packaging. A typical stemcell contains a bare minimum OS skeleton with a few common utilities pre-installed, a BOSH Agent, and a few configuration files to securely configure the OS by default. For example: with vSphere, the official stemcell for Ubuntu Trusty is an approximately 500MB VMDK file. With AWS, official stemcells are published as MIs that can be used in an AWS account. Stemcells do not contain any specific information about any software that will be installed once that stemcell becomes a specialized machine in the cluster; nor do they contain any sensitive information which would make them unable to be shared with other BOSH users. This clear separation between base OS and later-installed software is what makes stemcells a powerful concept. In addition to being generic, stemcells for one OS (e.g. all Ubuntu Trusty and Xenial stemcells) are exactly the same for all infrastructures. This property of stemcells allows BOSH users to quickly and reliably switch between different infrastructures without worrying about the differences between OS images. The CF BOSH team is responsible for producing and maintaining an official set of stemcells. Cloud Foundry currently supports Ubuntu Trusty and Xenial on vSphere, AWS, OpenStack, Google, and Azure infrastructures.

2. **WHAT IS STEMCELL HARDENING?** Stemcell hardening is the process of securing a stemcell by reducing its surface of vulnerability, which is larger when a system performs more functions; in principle a single-function system is more secure than a multipurpose one. There are various methods of hardening Linux systems. Common techniques include reducing available methods of attack by implementing more restrictive and/or conservative configurations of the OS kernel and system services, changing default passwords, the removal of unnecessary software, unnecessary usernames and logins, and the disabling or removal of unnecessary services.

3. **WHAT IS OUR GENERAL APPROACH TO STEMCELL HARDENING?** The CF stemcell is essentially a distinct Linux distribution. As such, industry-standard benchmarks are not entirely appropriate when assessing the security posture of the stemcell, but Pivotal has considered and incorporated hardening guidance from various sources both commercial and government. Some parts of the existing recommended industry-standard hardening configurations will certainly apply, but some other parts do not apply. In addition, because each stemcell is a unique Linux distribution, existing industry-standard benchmarks are silent on some important aspects of hardening the stemcell configurations. The following paragraphs describe the different categories of stemcell hardening configurations, and provide a count of the number of tests currently in each category. **Note:** The most current description of what has been delivered is always available in the BOSH public Pivotal Trackers.

   a. **Baseline Passing:** common hardening tests that pass without any changes to the stemcell or to test procedures.  *(130 tests)*
   
   b. **Test Amended:** Stemcells are optimized for cloud deployment and some configuration settings are not stored in traditionally-expected locations. The industry standard test was changed to conform with stemcell design to accurately check the recommended setting. This new test reflects the changes to the industry standard test but the stemcell adheres to commonly accepted guidance.  *(36 tests)*
   
   c. **Additional Hardening:** Configuration hardening improvements that have been made to the stemcell. As with most software, a stemcell’s security improves over time and every stemcell release is tested to ensure that it is suitable for use with its associated CF release. Later releases of a stemcell may include additional security features that were not present in earlier releases.  *(86 tests)*
   
   d. **New CF-specific Tests:** New tests that have been added to check CF stemcell-specific configurations. These tests are not yet part of any industry standard Ubuntu benchmark. This category of tests is still under development and additional tests will be added over time.  *(20 tests)*

4. **WHAT ARE THE MAJOR FOCUS AREAS FOR OUR STEMCELL HARDENING APPROACH?**

   a. **Maintenance, Updates, and Patching**

      i. Regular patches and feature enhancements are delivered via routine BOSH deployments of updated stemcells (obviates apt-get upgrade).

   b. **File System Hardening**

      i. The /tmp directory is configured to be on a separate partition.
      
      ii. Users cannot create character or block special devices in the /tmp filesystem.
      
      iii. Users cannot create set user id files in the /tmp filesystem.
      
      iv. Users cannot run executable binaries from the /tmp filesystem.
      
      v. The temporary storage directories such as /tmp and /var/tmp are mounted on a dedicated partition, and configured with appropriately limiting options such as nodev, nosuid, and noexec.
      
      vi. Each of the following directories is in a separate partition, with mount options managed via BOSH agent:

      - /var
      - /var/log
      - /var/log/audit
      - /home
vii. File system mount options for users’ home directories are limited via appropriate mount options including nodev.
viii. Removable media may not be mounted as character or block special device.
ix. Executable programs may not run from removable media.
x. setuid and setgid are not allowed on removable media.
xi. Users cannot create special devices in shared memory partitions.
xii. Users cannot put privileged programs onto shared memory partitions.
xiii. Users cannot execute programs from shared memory partitions.
xiv. Users cannot delete or rename files in world-writable directories such as /tmp that are owned by other users.
xv. Supplementary and exotic Linux file systems that are unused in CF have been disabled.
xvi. Automount of USB drives or disks is not permitted.

b. Boot Security
   i. The owner and group for the bootloader config (/boot/grub/grub.cfg) is set to root. Only root has read and write access to this file.
   ii. Boot loader has been configured so that a password is required to reboot the system.
   iii. Unauthorized users cannot reboot the system into single user mode.

d. Process Security
   i. Users cannot override the soft limit for core dumps.
   ii. Randomized virtual memory region placement is enabled.
   iii. Prelinking of shared libraries is disabled.

e. Minimization of Attack Surface
   i. The Network Information Service (“NIS”) is not used in CF and is not installed.
   ii. The Berkeley rsh-server package is not used in CF and is disabled.
   iii. Classic rsh-related tools are not used in CF and are not installed.
   iv. The following servers are not used on CF stemcells and are disabled:
      - talk server
      - telnet server
      - tftp-server
      - Avahi
      - print
      - DHCP
      - DNS
      - FTP
      - IMAP
      - POP
      - HTTP
      - SNMP
   v. The talk client is not used in CF and is not installed.
   vi. The Extended InterNET Daemon (xinetd) is not used in CF and is disabled.
   vii. The following network services are not used in CF and are disabled:
      - chargen
      - daytime
      - echo
      - discard
      - time
   viii. The X Window system is not used in CF and is not installed.
   ix. NTP time setting is synchronized on the stemcell via the ntpdate utility.
   x. The Samba daemon is not used in CF and is disabled.
   xi. The Mail Transfer Agents (MTA) process only local mail.
   xii. The rsync service is not used in CF and is disabled.
   xiii. The biosdevname tool is disabled.

f. Network Security
   i. IPv4 networking is configured such that IP forwarding is disabled.
   ii. The IPv4 networking has been configured such that the host cannot send ICMP redirects.
   iii. IPv4 networking has been configured such that the system does not accept source routed packets.
   iv. IPv4 networking is configured such that ICMP redirects are not accepted.
   v. ICMP echo and timestamp requests with broadcast or multicast destinations will be ignored.
vi. The stemcell will ignore malformed ICMP error responses.

vii. IPv4 networking is configured for source route validation.

viii. TCP SYN cookies are enabled.

ix. Stemcells are set to refuse IPv6 router advertisements.

x. The /etc/hosts.allow file exists and is empty.

xi. The /etc/hosts.allow and /etc/hosts.deny files are protected from unauthorized write access.

xii. The /etc/hosts.deny file exists and is empty.

xiii. The following protocols are not used in CF and are disabled:

- SCTP
- DCCP
- TIPC
- LDAP
- RDS

xiv. Wireless interfaces are disabled.

xv. IPv6 is not used in CF deployments and the IPv6 protocol is disabled.

g. Auditing

i. Audit log file size is configured for a manageable maximum size of 6 MB.

ii. The system auditd logs have been configured such that the system is resilient in the event of a denial of service attack on the auditd daemon.

iii. Auditd daemon is configured such that all auditd logs are kept after rotation.

iv. The auditd service is enabled.

v. Auditing of successful and failed login/logout events is enabled.

vi. The Linux auditing subsystem has been configured in accordance with best practice industry guidance to capture all security-relevant events. The /etc/audit/audit.rules configuration now contains more than 50 monitoring rules.

vii. Audit records are created for loading and unloading of kernel modules and for system calls.

viii. File Integrity Monitoring can be done on the stemcell (via a BOSH Add-on).

h. Authentication and Authorization

i. The cron daemon is enabled.

ii. Access to the /etc/crontab file is limited to root.

iii. Access to the cron utility configuration via the hourly, daily, weekly, and monthly directories is limited.

iv. User authorization to schedule cron jobs is limited.

v. Only the vcap user is whitelisted to use the cron and at utilities.

vi. Password requirements follow industry best practice guidance and enforce a minimum length of 14 characters, with at least one each of: digit, uppercase, lowercase and special characters.

vii. Password reuse: users cannot reuse their twenty most recent passwords.

viii. SSM protocol version is configured for SSH-2.

ix. Logging level for SSH event is INFO.

x. Minimum permissions are set on /etc/ssh/sshd_config.

xi. SSH X11 forwarding is disabled.

xii. The MaxAuthTries parameter for SSH is set to 3 attempts per connection.

xiii. SSH is configured to require passwords and ignore host-based authentication.

xiv. Root logins are not allowed over SSH.

xv. Users cannot set environment variables through the SSH daemon.

xvi. SSH has been configured to use strong ciphers:

- aes128-ctr
- aes192-ctr
- aes256-ctr

xvii. Idle SSH sessions are terminated after 15 minutes, and no client “keep alive” messages are sent.

xviii. Idle SSH sessions are terminated after 15 minutes. No client “keep alive” messages are sent.

xix. The SSH login banner may be configured to display site-specific text before user authentication is permitted (via BOSH Add-on).

xx. Root login is only permitted via console, not via tty devices.

xxi. Only the vcap user is authorized in the sudo group.

xxii. Only users in the root group (a.k.a. wheel) are authorized to run the su command.

i. Compliance

i. Contents of /etc/issue and /etc/issue.net have been configured to the phrase: “Unauthorized use is strictly prohibited. All access and activity is subject to logging and monitoring.” This may be amended if and as necessary via a BOSH Add-on.

ii. The Message of the Day file /etc/motd is not used, but may be populated via a BOSH Add-on if needed.

iii. Identification of the OS and/or version information about the OS does not appear in any login banners.
j. **File System Permissions**
   
   i. The `/etc/passwd`, `/etc/shadow`, and `/etc/group` files are protected from unauthorized write access.
   
   ii. Use and/or presence of any world-writable files has been audited, and minimized to the extent possible for CF.
   
   iii. By default, all stemcell files are owned by a known user and group, and may not belong to a non-existent user or group.
   
   iv. Use of SUID and GUID is restricted, and only the `/usr/bin/sudo` and `/bin/su` programs are authorized as SUID and/or GUID programs.

k. **User Account Management**
   
   i. Users cannot change their password more than once a day.
   
   ii. Users are notified 7 days before their passwords expire.
   
   iii. Interactive logins are disabled for system accounts.
   
   iv. The GID for the root account is 0.
   
   v. User accounts may not have empty passwords.
   
   vi. NIS is not used in CF, and integration of OS security configuration with legacy NIS permissioning is not enabled (e.g., for `/etc/passwd`, `/shadow`, and `/group`).
   
   vii. By default, the only UID 0 account present is root.
   
   viii. By default, the root PATH does not include any risky directory such as the current working (.) or any writable directory.
   
   ix. Minimum privileges are applied to all users' hidden configuration ("dot") files.
   
   x. The `.netrc` and `.rhosts` and `.forward` files are not used in CF and are not present in any user home directory.
   
   xi. Any group present in the `/etc/passwd` file must also exist in the `/etc/group` file.
   
   xii. Users defined in `/etc/password` must have a valid home directory.
   
   xiii. Users must own their home directories.
   
   xiv. All references to user and group names, as well as UID and/or GID identifiers, are self consistent, with no duplicates or orphans allowed.
   
   xv. By default, the shadow group is not used in CF and must be empty.
Windows Stemcell Hardening

This topic describes the security measures that Pivotal uses to harden the Windows 2016 stemcell.

Note: This document applies to Windows stemcell v1709.x.

Local Group Policy Settings

The Windows stemcell contains a version of Windows Server 2016 with a set of Local Group Policy settings optimized for security. These settings begin with the WS2016 Member Server Security Compliance v1.0 baseline, included in Microsoft Security Compliance Manager v4.0. For more information about this baseline, see Windows Security Baselines.

Pivotal has collaborated with Microsoft to further harden the stemcell by implementing Local Security Policies settings, according to the recommended security baseline defined in Microsoft Security Compliance Manager. The table below lists these overrides.

Note: Pivotal will continue to revise these settings as Microsoft releases updates.

<table>
<thead>
<tr>
<th>Name</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn off Automatic Download and Install of updates</td>
<td>Enabled</td>
</tr>
<tr>
<td>Allow Remote Shell Access</td>
<td>Disabled</td>
</tr>
<tr>
<td>Windows Firewall: Private: Display a notification</td>
<td>No</td>
</tr>
<tr>
<td>Windows Firewall: Domain: Display a notification</td>
<td>No</td>
</tr>
<tr>
<td>Windows Firewall: Public: Display a notification</td>
<td>No</td>
</tr>
<tr>
<td>Network access: Do not allow storage of passwords and credentials for network auth</td>
<td>Enabled</td>
</tr>
<tr>
<td>Access this computer from the network</td>
<td>Administrators</td>
</tr>
<tr>
<td>Deny log on as a batch job</td>
<td>Guests</td>
</tr>
<tr>
<td>Deny log on as a service</td>
<td>Guests</td>
</tr>
<tr>
<td>Deny log on through Remote Desktop Services</td>
<td>Guests</td>
</tr>
</tbody>
</table>
Certificates and TLS in PCF

This section provides links to topics about certificate and TLS infrastructure in Pivotal Cloud Foundry (PCF).

- **TLS Connections in PCF**: Provides information on TLS is used in PCF deployments, including supported TLS cipher-suites.
- **Securing Traffic into Cloud Foundry**: Provides instructions on how to configure Transport Layer Security (TLS) termination for HTTP traffic into PCF.
- **Providing a Certificate for Your TLS Termination Point**: Describes how to configure TLS certificates for Pivotal Application Service (PAS).
- **Managing Certificates with the Ops Manager API**: Describes how to manage internal certificate authorities (CAs) and certificates in PCF that are visible to the Ops Manager API.
- **Rotating Certificates**: Describes how to rotate internal CAs and certificates in PCF.
- **Adding a Custom Certificate Authority**: Describes how to add a custom CA to issue digital certificates in a Pivotal Cloud Foundry (PCF) deployment
- **Trusted System Certificates**: Discusses where apps deployed to PAS can find trusted system certificates.
TLS Connections in PCF

This topic classifies the different paths through which external clients, internal components, app containers, and app services communicate in Pivotal Cloud Foundry (PCF), and how the platform uses Transport Layer Security (TLS) protocols to secure these communications.

Types of Network Communication in PCF

This section classifies the different types of network communication in PCF and how they are secured with TLS.

Within a PCF deployment, TLS secures connections between components like the BOSH Director and service tiles. PCF components also use TLS connections to secure communications with external hardware, such as customer load balancers.

Between an External Client and an App

The following diagram illustrates the flow of communication from a client making a request to an app:

The following table describes each component involved in receiving a client request and where their certificates for TLS termination originate.

<table>
<thead>
<tr>
<th>Component</th>
<th>Certificate Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Load Balancer</td>
<td>Enterprise Root CA.</td>
</tr>
<tr>
<td>Gorouter</td>
<td>Enterprise Root CA.</td>
</tr>
<tr>
<td>App</td>
<td>PCF root CA dedicated to app instance identity. For more information, see App Instance Container Identity Credentials.</td>
</tr>
</tbody>
</table>

Between Platform Components
The following diagram illustrates communication between platform components, secured with TLS.

The CredHub instance in BOSH generates certificates for all components in PCF. The certificates are self signed by default. To issue certificates signed by your enterprise, you can add a custom CA to CredHub.

**Component Communication Details**

The following topics list the paths, ports, and protocols that subsystems within Pivotal Application Service (PAS) use to communicate.

- Cloud Controller Network Communications
- Diego Network Communications
- Loggregator Network Communications
- MySQL Network Communications
- NATS Network Communications
- Routing Network Communications
- UAA Network Communications

**Between Apps**

The following diagram illustrates TLS communications between apps running on PCF. In this example, the `frontend-blue` and `frontend-green` apps both receive client requests, but only the `frontend-green` app is allowed to communicate with the `backend` app.
Apps can communicate with each other over TLS using certificates generated by a PCF root CA dedicated to app instance identity. For more information, see [App Instance Container Identity Credentials](#).

Developers specify which apps are allowed to communicate using container networking policies. For more information, see [Configuring Container-to-Container Networking](#).

### Between Apps and On-Platform Services

The following diagram illustrates TLS communication between apps and managed, on-platform services. It uses Pivotal Cloud Cache as an example of a managed service.
BOSH CredHub issues certificates to the Pivotal Cloud Cache components. For the app, the developer must retrieve a copy of this certificate using the CredHub API and place it in the truststore for the app. For more information, see the Developing an App Under TLS document in the Pivotal Cloud Cache documentation.

Separately, PAS Runtime Credhub might store credentials for the app to access a service over the TLS connection, adding a second layer of security. For more information, see Securing Services Instance Credentials with Runtime CredHub.

Between Apps and External Services

The following diagram illustrates communications between apps and external, brokered services secured with TLS.
The developer must retrieve the certificate from the external service and provide it to their app. One way to do this is by placing the certificate in Runtime CredHub and modifying your app to consume the certificate through the CredHub Service Broker for PCF.

**App Instance Container Identity Credentials**

Each app instance container in PCF has its own identity credentials. This section is meant to help PCF operators and developers understand and use these credentials.

**About App Instance Identity Credentials**

See the following table to learn about app instance identity credentials.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>• For app developers to enable secure TLS communications from their apps.</td>
</tr>
<tr>
<td></td>
<td>• For PCF to use internally to validate the identities of app instances.</td>
</tr>
<tr>
<td>Type</td>
<td>• A PEM-encoded X.509 certificate and PKCS #1 RSA private key.</td>
</tr>
<tr>
<td>Location</td>
<td>• PCF presents the certificate and private key to the app instance through the container filesystem.</td>
</tr>
<tr>
<td>Properties of certificate</td>
<td>• The Common Name is the app instance GUID.</td>
</tr>
<tr>
<td></td>
<td>• The Subject of the certificate contains an Organizational Unit in the form of app:APP-GUID.</td>
</tr>
<tr>
<td></td>
<td>• The certificate contains a Subject Alternative Name (SAN) with the IP address for the app instance container.</td>
</tr>
<tr>
<td></td>
<td>• The certificate is valid for 24 hours after being issued.</td>
</tr>
<tr>
<td>Contents of certificate file</td>
<td>• A chain of PEM-encoded certificates, with the instance-specific certificate first in the list and any intermediates following it.</td>
</tr>
<tr>
<td></td>
<td>• PCF includes a root Certificate Authority (CA) dedicated to app instance identity. This CA is saved in the system trust.</td>
</tr>
</tbody>
</table>
Using the Credentials

If you want to enable secure TLS communications from an app using container instance identity credentials, ensure that you do the following:

- **Add the credentials to your development stack configuration:**
  - The credentials must be present in your development stack configuration for your app to use them. You can retrieve the credentials through following environment variables, which PCF sets to the locations of key and certificate files.

<table>
<thead>
<tr>
<th>Credential / Keypair Element</th>
<th>Environment Variable</th>
<th>Command to Retrieve Credential Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate Chain</td>
<td>CF_INSTANCE_CERT</td>
<td><code>cf ssh APP-NAME -c 'cat $CF_INSTANCE_CERT'</code></td>
</tr>
<tr>
<td>Private Key</td>
<td>CF_INSTANCE_KEY</td>
<td><code>cf ssh APP-NAME -c 'cat $CF_INSTANCE_KEY'</code></td>
</tr>
</tbody>
</table>

- **Reload the credential files before they expire:**
  - PCF rotates the credentials shortly before the current certificate expires. Apps that use these credentials must reload the certificate and key file contents either periodically or in reaction to filesystem watcher events.

- **Configure external clients or servers to trust the root CA:**
  - To enable secure TLS communication between an app and a client or server external to PCF, you must configure the external client or server to trust the CA that issues app instance container identity credentials. See the **Issuing Authority** row of the table in [About App Instance Identity Credentials](#).

**Additional Information**

For more information about instance identity credentials, see the [Instance Identity](#) document in the diego-release repository.

**TLS Cipher Suites**

By default, PCF uses a limited set of cipher suites to secure its internal communications. However, some components used in PCF, like the Gorouter and HAProxy, may support additional TLS cipher suites to accommodate older clients outside of PCF.

- **The AWS Classic load balancer does not support the recommended TLS cipher suites. See [Securing Traffic into Cloud Foundry](#) for details and mitigations.**

For components that allow you to configure TLS cipher suites, only specify the TLS cipher suites that you need.

**TLS Cipher Suite Recommendations**

The default and recommended version of TLS to use is TLS v1.2.

The recommended TLS cipher suites to use within PCF are the following:

- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`

**Gorouter Configuration**

As part of your PAS networking configuration, you must specify the TLS cipher suites that the Gorouter uses to secure its communications. Only specify the cipher suites that you need.

The recommended TLS cipher suites for the Gorouter are:
HAProxy Configuration

As part of your PAS networking configuration, you must specify the TLS cipher suites that HAProxy uses to secure its communications. Only specify the cipher suites that you need.

The recommended TLS cipher suites for HAProxy are:

- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES256-GCM-SHA384

You can specify other cipher suites and a different minimum version of TLS support if your deployment requires it. For a list of other cipher suites and other versions of TLS that are optionally supported by HAProxy, see `ciphers - Cipher Suite Names` in the OpenSSL documentation.

If you use the default and recommended Gorouter TLS cipher suites in PAS, then ensure you have included these Gorouter TLS cipher suites in your HAProxy TLS cipher suite configuration.

If you change the default Gorouter TLS cipher suites in PAS, and you change the TLS cipher suites for HAProxy, ensure that you have at least one overlapping TLS cipher suite within the two sets.

For instructions on how to configure the TLS cipher suites for HAProxy, see the PAS installation documentation for the IaaS of your deployment. For example, if you are deploying PAS on GCP, see Configuring PAS.
Custom Certificate Authorities

This topic provides an overview of using custom certificate authorities (CAs) in a Pivotal Cloud Foundry (PCF) deployment.

Overview

To secure traffic in your PCF deployment, you must provide a CA to issue digital certificates. This can be either a Pivotal-generated or custom CA. When you add and activate a new CA, a digital certificate is issued to BOSH Director. BOSH Director then passes the certificate to other components in your PCF deployment.

Pivotal recommends you supply a CA from a trusted provider when using a production environment. While you can create your own custom CAs if necessary, a trusted CA is more secure because it has been authenticated by the trusted entities permitted to issue them.

Note: Elliptic Curve Digital Signature Algorithm (ECDSA) certificates are not supported in PCF.

Add a Custom CA

You can add a new custom CA as part of the procedure for rotating CAs and other certificate types in PCF. To add and activate a new custom CA in PCF, see Rotate Root and Leaf Certificates.
Managing Certificates with the Ops Manager API

This topic describes how to manage and retrieve information about certificates in PCF using the Ops Manager API.

Overview

The Ops Manager API includes endpoints for managing and retrieving information about certificates in a PCF deployment.

For more information about Ops Manager API endpoints for managing certificates, see Certificate Authorities in the Ops Manager API documentation.

Prerequisites

To use the Ops Manager API, you must generate an access token by authenticating with the Ops Manager User Account and Authentication (UAA) server.

For more information about authenticating with UAA, see Using Ops Manager API.

Generate a Single RSA Certificate

To generate and return a new RSA certificate signed by the root certificate authority (CA), use `curl` to make the following API call:

```bash
curl "https://OPS-MAN-FQDN/api/v0/certificates/generate\ -X POST\ -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

Where `YOUR-UAA-ACCESS-TOKEN` is your Ops Manager access token without any newline characters such as `\n`.

Retrieve the Ops Manager Root CA

You can view the Ops Manager root CA as a file or in JSON format.

Retrieve the Ops Manager Root CA as a File

To return the Ops Manager root CA as a file, use `curl` to make the following API call:

```bash
```

Where `YOUR-UAA-ACCESS-TOKEN` is your Ops Manager access token without any newline characters such as `\n`.

Retrieve the Ops Manager Root CA as JSON

To return the Ops Manager root CA as JSON, use `curl` to make the following API call:

```bash
```

Where `YOUR-UAA-ACCESS-TOKEN` is your Ops Manager access token without any newline characters such as `\n`.

List all RSA Certificates
To return metadata from all deployed RSA certificates visible to Ops Manager, except the root CAs, use `curl` to make the following API call:

```
curl *https://OPS-MAN-FQDN/api/v0/deployed/certificates
   -X GET
   -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

Where `YOUR-UAA-ACCESS-TOKEN` is your Ops Manager access token without any newline characters such as `\n`.
Rotating Certificates

This topic describes how to rotate the root certificate authorities (CAs) and leaf certificates in Pivotal Cloud Foundry (PCF) that are visible to the Ops Manager API.

This topic includes the following:

- A list of the different types of certificates that require planned rotation. See Certificate Types.
- Instructions for checking the expiration date of and rotating the Ops Manager root CAs and leaf certificates. See Master Procedure: Check and Rotate Certificates.

For information about rotating IPsec certificates, see Rotating IPsec Certificates.

For information about using trusted third-party certificates for both apps hosted on PCF and internal PCF components, see Setting Trusted Certificates.

Overview

The Ops Manager API manages and lists internal certificates that enable PCF components to communicate with each other securely using TLS. It can also list certificates used externally, such as SAML certificates that authenticate to an external identity provider.

To keep PCF running, you must keep track of which certificates are set to expire soon, and rotate them before they expire. To do this, follow the instructions in Master Procedure: Check and Rotate Certificates.

Certificate Types

PCF uses a root CA and various leaf certificates. Root CAs are self-signed certificates that issue leaf certificates. Root CAs can be generated by Pivotal or custom.

Leaf certificates are signed by a CA and are used to identify resources in PCF. Both root CAs and leaf certificates require planned rotation in PCF.

The following types of PCF certificates require planned rotation:

- **Ops Manager Root CA**: The Ops Manager root CA issues other certificates that PCF uses. The root CA can be a Pivotal-generated CA or your own custom CA. For more information about viewing the root CAs for Ops Manager, see Listing the Root Certificate Authorities.

- **Non-configurable Certificates**: Non-configurable certificates are leaf certificates either created by a CA stored in Ops Manager, or created and stored by CredHub and managed by Ops Manager calls to the CredHub API. Non-configurable certificates are issued directly by the Ops Manager root CA, or by intermediate CAs in a chain of trust originated by the root CA. For more information about viewing non-configurable leaf certificates, see Getting Information About Certificates for Products. For more information about generating non-configurable leaf certificates, see Generating New Certificates.

- **Configurable Certificates**: Configurable certificates are leaf certificates supplied by the user and pasted into configuration fields in Ops Manager. Some configuration panes include a Generate RSA Certificate button that supplies valid certificates, but users can obtain configurable certificates from elsewhere. For more information about viewing configurable leaf certificates, see Getting Information About Certificates for Products.

- **Non-rotatable Certificates**: Non-rotatable certificates are leaf certificates that, like non-configurable certificates, are issued by the root CA. Unlike non-configurable certificates, non-rotatable certificates cannot be rotated by the Ops Manager API. For more information about viewing non-rotatable leaf certificates, see Getting Information About Certificates for Products.

In addition to the types of certificates listed above, some Pivotal products issue their own tile certificates that are not managed by or visible to the Ops Manager API. These tile certificates do not require planned rotation because they rotate automatically with product upgrades.

Pivotal Application Service (PAS) and Pivotal Container Service (PKS) both use tile certificates in addition to their Ops Manager certificates.

Master Procedure: Check and Rotate Certificates

The following master procedure checks expiration dates of different types of internal certificates and rotates them only as necessary. You can run this procedure only when records show that your certificates will expire soon, or else periodically to comply your organization’s security compliance policies.

To check and rotate certificates, do the following:

© Copyright Pivotal Software Inc, 2013-2019 1264
1. Follow the Check Ops Manager Root CA Expiration Date procedure and record your root CA expiration date.

2. Follow the Check Leaf Certificate Expiration Dates procedure and record any leaf certificates that are expiring soon.

3. Follow the Identify Non-Configurable, Configurable, and Unrotatable Leaf Certificates to determine which types of leaf certificates require rotation.

4. Do the following to rotate certificates as necessary:
   a. If you have any non-rotatable certificates expiring soon, call Pivotal Support.
   b. If your root CA expires soon, follow the Rotate Root and Leaf Certificates procedure.
   c. If you have non-configurable leaf certificates expiring soon, but not your root CA, follow the Rotate Non-Configurable Certificates procedure.
   d. If you have configurable leaf certificates expiring soon, but not your root CA, follow the Rotate Configurable Certificates procedure.

Check Expiration Dates and Certificate Types

Complete the following procedures to check the expiration dates and types of CAs and leaf certificates that the Ops Manager API lists and manages.

Check Ops Manager Root CA Expiration Date

This procedure describes how to check the expiration date for the Ops Manager root CA. The Ops Manager root CA expires four years after creation.

To check the Ops Manager root CA expiration date, do the following:

1. Perform the steps in the Using Ops Manager API topic to target and authenticate with the Ops Manager User Account and Authentication (UAA) server. Record your Ops Manager access token, and use it for YOUR-UAA-ACCESS-TOKEN in the steps below.

   Note: When you record your Ops Manager access token, remove any newline characters such as \n.

2. To retrieve the Ops Manager root CA, use curl to make an Ops Manager API call to the https://OPS-MAN-FQDN/api/v0/certificateAuthorities endpoint. For example:

   ```
   curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities"
   -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
   ```

3. To make the JSON output more readable, you can pipe it to jq or another text editor with JSON formatting.

4. In the certificateAuthorities list returned, if there is more than one, find the CA with "active": true.

5. To determine its expiration date of the active CA, refer to its expiresOn value. For example, the root CA shown below expires on September 5, 2019:

```json
{
   "certificateAuthorities": [
      {
         "guid": "9c9a110c8f82a1e4aaca",
         "issuer": "Pivotal",
         "createdOn": "2017-09-05T22:47:53Z",
         "active": true,
         "certPem": "-----BEGIN CERTIFICATE-----
[...

```

Check Leaf Certificate Expiration Dates

This procedure describes how to check the expiration dates of non-configurable and configurable leaf certificates. Non-configurable leaf certificates expire after two years. Configurable leaf certificates generated by Ops Manager also typically expire after two years.

To check the expiration dates for non-configurable and configurable leaf certificates, do the following:
1. If you haven’t already, perform the steps in the Using Ops Manager API topic to target and authenticate with the Ops Manager User Account and Authentication (UAA) server. Record your Ops Manager access token, and use it for \texttt{YOUR-UAA-ACCESS-TOKEN} in the steps below.

   \begin{itemize}
   \item **Note:** When you record your Ops Manager access token, remove any newline characters such as \\
   \end{itemize}

2. To check the system for certificates that expire within a given time interval, use \texttt{curl} to call the \texttt{https://OPS-MAN-FQDN/api/v0/deployed/certificates?expires_within=\text{\texttt{TIME}}} API endpoint, replacing \texttt{\text{\texttt{TIME}}} with an integer-letter code. Valid letter codes are \texttt{d} for days, \texttt{w} for weeks, \texttt{m} for months, and \texttt{y} for years.

   For example, the following command searches for certificates expiring within six months:

   \begin{verbatim}
   curl "https://OPS-MAN-FQDN/api/v0/deployed/certificates?expires_within=6m" \\
   -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
   \end{verbatim}

   Replace \texttt{YOUR-UAA-ACCESS-TOKEN} with the \texttt{access_token} value you recorded in the previous step.

3. To make the JSON output more readable, pipe it to \texttt{jq} or another text editor with JSON formatting.

4. To determine the expiration date of each certificate, refer to its \texttt{expires_on} value.

5. To determine the type of each certificate in the output, follow the Identify Non-Configurable, Configurable, and Unrotatable Leaf Certificates procedure.

**Identify Non-Configurable, Configurable, and Unrotatable Leaf Certificates**

Output from the \texttt{deployed/certificates} endpoint, such as the output generated in the Check Leaf CertificateExpiration Dates procedure, combines information about non-configurable, configurable, and unrotatable leaf certificates into a single list.

You need to manually rotate the different leaf certificate types in different ways.

The following rules identify the type of each leaf certificate requiring rotation:

- **Non-Rotatable Certificates:** Non-Rotatable leaf certificates have the following property value:
  \begin{itemize}
  \item \texttt{variable_path} is \texttt{/opsmgr/bosh_dns/tls_ca}
  \end{itemize}

  If you have non-rotatable certificates expiring soon, call Pivotal Support \texttt{.}

- **Non-Configurable Certificates:** Non-Configurable leaf certificates have the following property values, but are not non-rotatable as identified above:
  \begin{itemize}
  \item \texttt{configurable} is \texttt{false}
  \item \texttt{location} is either \texttt{ops_manager} or \texttt{credhub}
  \end{itemize}

- **Configurable Certificates:** Configurable leaf certificates have the following property value:
  \begin{itemize}
  \item \texttt{configurable} is \texttt{true}
  \end{itemize}

**Rotate CAs and Leaf Certificates**

The following procedures rotate CAs and leaf certificates that are listed or managed by the Ops Manager API.

\begin{itemize}
\item **Note:** The rotation procedures described in this topic does not work when your certificates have already expired. If your certificates have expired, contact Pivotal Support \texttt{.}
\end{itemize}

**Rotate Root and Leaf Certificates**

This procedure uses the Ops Manager API to rotate the Ops Manager root CA and non-configurable leaf certificates.

Rotating the Ops Manager root CA automatically rotates all configurable leaf certificates. You can also rotate configurable leaf certificates separately from rotating the root CA. For information about rotating configurable leaf certificates without also rotating the root CA, see Rotate Configurable Certificates.

PCF users never need to manually rotate intermediate CAs, because they rotate automatically when the root CA is rotated.
To prevent system downtime, this procedure includes two BOSH redeploy. When you click Apply Changes for the first time, BOSH applies new certificates to PCF components alongside the old ones. The second Apply Changes then deletes the old certificates. Each successful redeploy verifies that the certificate rotation process is proceeding correctly.

⚠️ warning: You must complete these steps in the exact order specified. Otherwise, you may damage your deployment.

**Step 1: Add a New Root CA**

Follow this procedure to add a new root CA for Ops Manager. The new root CA can be a Pivotal-generated CA or your own custom CA.

Note: This procedure regenerates the NATS CA in unison with the Ops Manager root CA.

To add a new root CA for Ops Manager, do the following:

1. If you haven’t already, perform the steps in the Using Ops Manager API topic to target and authenticate with the Ops Manager User Account and Authentication (UAA) server. Record your Ops Manager access token, and use it for `YOUR-UAA-ACCESS-TOKEN` in the following procedures.

   Note: When you record your Ops Manager access token, remove any newline characters such as ' \n '.

2. Use Ops Manager to generate a new CA, or else add your own custom CA.

   Note: Elliptic Curve Digital Signature Algorithm (ECDSA) certificates are not supported in PCF.

   - To use your own custom CA, call the Ops Manager API `generate` endpoint as follows:

     ```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" 
  -X POST 
  -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" 
  -H "Content-Type: application/json" 
  -d '{"cert_pem": "-----BEGIN CERTIFICATE-----\YOUR-CERTIFICATE", "private_key_pem": "-----BEGIN RSA PRIVATE KEY-----\YOUR-KEY"}'
     ```

     Where:

     - `YOUR-CERTIFICATE` is your custom CA.
     - `YOUR-KEY` is your RSA key.
     - `YOUR-UAA-ACCESS-TOKEN` is your UAA access token.

     If the command succeeds, the API returns a response that includes the new CA certificate:

     ```json
     HTTP/1.1 200 OK
     {
      "certificate_authorities": [ 
        {
          "guid": "f7bc18f34f2a7a9403c3",
          "issuer": "YOUR-CA",
          "created_on": "2017-01-09",
          "expires_on": "2021-01-09",
          "active": true,
          "cert_pem": "-----BEGIN CERTIFICATE-----\nMIIC+zCCAeOgAwIBAgI....etc"
        }
      ]
    }
     ```

   - To use a Pivotal-generated CA, call the Ops Manager API `generate` endpoint as follows:

     ```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/generate" 
  -X POST 
  -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" 
  -H "Content-Type: application/json" 
  -d '{}'
     ```

     Where `YOUR-UAA-ACCESS-TOKEN` is your UAA access token.

     If the command succeeds, the API returns a response that includes the new CA certificate:

© Copyright Pivotal Software Inc, 2013-2019
3. Confirm that your new CA has been added by listing all of the root CAs for Ops Manager:


The API call returns the GUID of the newly added CA. For example:

```json
HTTP/1.1 200 OK

"certificateAuthorities": [
  {
    "guid": "f7bc18f34f2a7a9403c3",
    "issuer": "Pivotal",
    "createdOn": "2017-01-19",
    "expiresOn": "2021-01-19",
    "active": false,
    "certPem": "-----BEGIN EXAMPLE CERTIFICATE-----
MIIC+zCCAeOgAwIBAgIBADANBgkqhkiG9w0BAQsFADAQMBgNVBStrMBIGCCsGAQEB
-----END EXAMPLE CERTIFICATE-----"
  },
  {
    "guid": "a8ee01e33e3e3e3303e3",
    "issuer": "Pivotal",
    "createdOn": "2017-04-09",
    "expiresOn": "2021-04-09",
    "active": false,
    "certPem": "-----BEGIN CERTIFICATE-----
BBBC+eAAAe1gAwAAAeZ.....etc"
  }
]
```

Identify your newly added CA, which has **active** set to **false**. Record its GUID.

4. Navigate to [https://OPS-MAN-FQDN](https://OPS-MAN-FQDN) in a browser and log in to Ops Manager.

5. Click the BOSH Director tile in the Installation Dashboard.

6. Select the Director Config pane.

7. Select Recreate All VMs. This propagates the new CA to all VMs to prevent downtime.

8. Go back to the Installation Dashboard. For each service tile you have installed, do the following:
   a. Click the tile.
   b. Click the Errands tab.
   c. Enable the Recreate All Service Instances errand if provided. If you do not see this errand, contact [Pivotal Support](mailto:support@pivotal.io).

9. Click Review Pending Changes, then Apply Changes. When the deploy finishes, continue to the next section.
Step 2: Activate the New CA

To activate the new root CA, do the following:

1. Use `curl` to make an Ops Manager API call that activates the new CA, replacing `{CERT-GUID}` with the GUID of your CA that you retrieved in the previous section:

   ```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/{CERT-GUID}/activate" 
   -X POST 
   -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" 
   -H "Content-Type: application/json" 
   -d '{}'
   
   The API returns a successful response:

   HTTP/1.1 200 OK

2. List your root CAs again to confirm that the new CA is active:

   ```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" 
   -X GET 
   -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
   
   Examine the response to ensure that your new CA has `active` set to `true`.

Step 3 (Optional): Rotate Configurable Certificates

If you have configurable certificates expiring soon, complete the Rotate Configurable Certificates procedure. This procedure ensures that your next deploy includes both new configurable certificates and new non-configurable certificates. You can also rotate your configurable certificates later and perform an additional deploy.

Step 4: Rotate Non-Configurable Certificates from the New Root

To rotate non-configurable certificates from the new root CA, do the following:

1. Use `curl` to make an API call to regenerate all non-configurable certificates and apply the new CA to your existing BOSH Director:

   ```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/active/regenerate" 
   -X POST 
   -H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" 
   -H "Content-Type: application/json" 
   -d '{}'
   
   The API returns a successful response:

   HTTP/1.1 200 OK

2. Click the BOSH Director tile in the Installation Dashboard.

3. Select the Director Config pane.

4. Select Recreate All VMs. This propagates the new CA to all VMs to prevent downtime.

5. Go back to the Installation Dashboard. For each service tile you have installed, do the following:
   a. Click the tile.
   b. Click the Errands tab.
   c. Enable the Recreate All Service Instances errand if provided. If you do not see this errand, contact Pivotal Support.

6. Navigate to Ops Manager, click Review Pending Changes, and click Apply Changes to perform a second redeploy.

Step 5: (Optional) Delete the Old CA

⚠️ warning: Be sure to include the Review Pending Changes and Apply Changes from Step 4: Rotate Non-Configurable Certificates from the New.
If you want to delete the old CA, do the following:

1. List your root CAs to retrieve the GUID of your old, inactive CA:

```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities" \n-X GET \n-H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

2. Use `curl` to make an API call to delete your old CA, replacing `OLD-CERT-GUID` with the GUID of your old, inactive CA:

```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/OLD-CERT-GUID" \n-X DELETE \n-H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN"
```

The API returns a successful response:

```
HTTP/1.1 200 OK
```

3. Navigate to Ops Manager, click Review Pending Changes, and click Apply Changes.

---

**Rotate Non-Configurable Certificates**

This procedure regenerates non-configurable leaf certificates visible to the Ops Manager API, whether they are managed and stored by Ops Manager directly, or by CredHub at Ops Manager request.

Run by itself, this procedure does not rotate or otherwise affect the Ops Manager root CA.

To rotate non-configurable certificates, do the following:

1. Use `curl` to make an API call to regenerate all non-configurable certificates and apply the new CA to your existing BOSH Director:

```bash
curl "https://OPS-MAN-FQDN/api/v0/certificate_authorities/active/regenerate" \n-X POST \n-H "Authorization: Bearer YOUR-UAA-ACCESS-TOKEN" \n-H "Content-Type: application/json" \n-d '{}'
```

The API returns a successful response:

```
HTTP/1.1 200 OK
```

2. Navigate to Ops Manager, click Review Pending Changes, and click Apply Changes to perform a second redeploy.

---

**Rotate Configurable Certificates**

Configurable certificates are generated by the user and pasted into Ops Manager configuration panes where needed. Examples include certificates that terminate SSL traffic into PAS, or authenticate a Single Sign-On (SSO) service plan to an external SAML server.

For specific instructions on how to rotate SAML certificates for both PAS and the SSO service, see Identity Provider SAML Certificates. These certificates expire every two years, and every IDP has its own certificate that may require its own rotation cadence.

For Ops Manager, PAS, and other runtimes, Pivotal recommends only rotating configurable certificates within the more inclusive Rotate Root and Leaf Certificates procedure. But if you are sure that only your configurable certificates need rotation, and no others, you can run this procedure by itself and click Apply Changes at the end.

To rotate configurable certificates, do the following:

1. If you haven’t already, use the Ops Manager API deployed/certificates endpoint to retrieve information about your expiring configurable certificates, as described in the procedures Check Leaf Certificate Expiration Dates and Identify Non-Configurable, Configurable, and Unrotatable Leaf Certificates.

   The information for each configurable certificate looks like this:
For each configurable certificate that expires soon:

a. Find the text field the certificate is configured within the Ops Manager interface.
   - The `product_guid` field in the API output can help identify which tile the certificate is configured in. For example, the prefix `p-bosh-` refers to the BOSH Director tile, and the prefix `cf-` refers to the PAS tile.
   - The `property_reference` field in the API output can often help identify which Settings pane the certificate is configured in. For example, the `uaa.service_provider_key_credentials` property is configured in the PAS tile > UAA pane.
   - You might have to look through multiple configuration panes to identify where a certificate is configured.

b. Paste a new value for the certificate into the field

c. Click Save at the bottom of each pane in which you have provided new certificates.

3. If you are rotating configurable certificates within the Rotate Root and Leaf Certificates procedure, continue to the next step. Otherwise, if you are rotating configurable certificates only, return to the Installation Dashboard, click Review Pending Changes, and click Apply Changes.

Identity Provider SAML Certificates

SAML service provider credentials are one example of configurable certificates in PCF. When PAS is configured to use SAML as an identity provider, it uses a configurable CA certificate to authenticate to an external SAML server, by generating ephemeral certificates that PAS includes in its outbound request message headers. This CA has a two-year expiration period.

In addition, the SSO service shares the use of PAS SAML certificates for every SAML external Identity Provider (IdP) integration, such as trust, partnership, or Federation. You must rotate these in lockstep with PAS.

The Rotate Your SAML CA for SSO procedure below provides an example of how to rotate certificates for each IdP, including temporarily disabling certificate validation on the IdP side during the rotation.

The Knowledge Base article PCF Advisory - SAML Service Provider Credential Certificates Expire after 2 Years provides more information about rotating SAML certificates.

Rotate Your SAML CA for PAS and the SSO Service

SAML service provider credentials are only required for your PAS deployment if all of these conditions are met:

- You are using SSO in production for login to PAS or using the SSO service for login to apps.
- You are using SAML identity providers for PAS or SSO service plans.
- You had Ops Manager generate a certificate for you by clicking the Generate RSA Certificate button.
- You are validating the signature of SAML authentication request with your identity provider.

To regenerate and rotate SAML service provider certificates without disrupting PAS or your apps using the SSO service, do the following:

1. Disable certificate validation in your identity provider.

2. For PAS, follow the procedure in the table below that corresponds to your use case. This includes downloading and importing a new certificate and updated SAML metadata in your identity provider.

<table>
<thead>
<tr>
<th>Solution Name</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA Single Sign-On aka CA SiteMinder</td>
<td>Configuring CA as an Identity Provider</td>
</tr>
<tr>
<td>Ping Federate</td>
<td>Configuring PingFederate as an Identity Provider</td>
</tr>
<tr>
<td>Active Directory Federation Services</td>
<td>Configuring ADFS as an Identity Provider</td>
</tr>
</tbody>
</table>

3. For the SSO service, follow the procedure in the table below that corresponds to your use case. This includes downloading the SAML Service Provider metadata for each SAML identity provider integration, such as trust, partnership, or Federation, and importing the updated SAML Service Provider metadata in your identity provider.
<table>
<thead>
<tr>
<th>Solution Name</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADFS</td>
<td>Configuring a Single Sign-On Service Provider</td>
</tr>
<tr>
<td>CA SSO</td>
<td>Configuring a Single Sign-On Service Provider</td>
</tr>
<tr>
<td>Okta</td>
<td>Configure Okta as an Identity Provider</td>
</tr>
<tr>
<td>PingFederate</td>
<td>Configure PingFederate as an Identity Provider</td>
</tr>
<tr>
<td>Additional Documentation</td>
<td>Integration Guides</td>
</tr>
</tbody>
</table>

4. Re-enable certificate validation in your identity provider.
A Cloud Foundry Administrator can deploy a set of trusted system certificates. These trusted certificates are available in Linux-based app instances running on the Diego backend. Such instances include buildpack-based apps using the cflinuxfs3 stack and Docker-image-based apps.

If the administrator configures these certificates, they are available inside the instance containers as files with extension .crt in the read-only /etc/cf-system-certificates directory.

For cflinuxfs3-based apps, these certificates are also installed directly in the /etc/ssl/certs directory, and are available automatically to libraries such as openssl that respect that trust store. If the administrator configure these certificates, the location of the certificates is provided in the environment variable CF_SYSTEM_CERT_PATH on the instance container.
Disk Encryption

This topic describes how to secure Pivotal Cloud Foundry (PCF) VMs by encrypting their disks or rotating their disk encryption keys.

Introduction

Disk encryption protects data integrity if computing resources are stolen physically.

Disk encryption for VMs works at the IaaS level. An IaaS encrypts disks when it first creates them, or re-encrypts them when it rotates encryption keys. To encrypt disks in PCF, you must:

1. Configure the IaaS to encrypt disks when it creates them or re-creates them.
2. Trigger BOSH to re-create the existing VMs that use the disks, and create encrypted disks from now on for new VMs.

The procedures below detail how to do this for each IaaS.

Disks You Can Encrypt on a PCF VM

- The root file system for the VM. For BOSH-created VMs, this comes from the stemcell.
- Ephemeral disk for the VM.
- Persistent disk for the VM.

Which VMs Each Procedure Encrypts

For each IaaS, there are two disk encryption procedures, which encrypt different VMs:

- The BOSH Director procedure encrypts the disks used by the BOSH Director VM when you first create a PCF environment.
- The BOSH-Deployed VM procedure encrypts disks for the VMs that the BOSH Director creates, after BOSH has been deployed.

Encrypt Disks or Rotate Keys

You can use the same procedure to either encrypt disks for the first time or rotate encryption keys.

For BOSH-deployed VMs, some IaaSes let you associate a policy with the BOSH process that automatically encrypts all disks BOSH creates. On AWS, BOSH must explicitly tell the IaaS to encrypt each disk that it creates, and passes in an encryption key. The following table summarizes these differences:

<table>
<thead>
<tr>
<th>IaaS</th>
<th>How configured</th>
<th>How encrypted</th>
<th>User can supply key</th>
<th>BOSH stores key ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>User pastes key ARN (ID) into Ops Manager</td>
<td>BOSH tells IaaS to encrypt disks it creates</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Azure (with Managed Disks)</td>
<td>User configures IaaS to associate encrypt policy with BOSH</td>
<td>IaaS automatically encrypts disks it creates for BOSH</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Azure (with Storage Accounts)</td>
<td>User configures IaaS to associate encrypt policy with BOSH</td>
<td>IaaS automatically encrypts disks it creates for BOSH</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>vSphere</td>
<td>User configures IaaS to associate encrypt policy with BOSH</td>
<td>IaaS automatically encrypts disks it creates for BOSH</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Azure

Azure provides virtual disk space through Azure Storage accounts. In some regions, Azure offers a Managed Disks service for storage accounts, which allocates disk space flexibly on demand.
Managed Disks vs Unmanaged Storage Accounts

For disk encryption, Pivotal recommends Managed Disks storage where available. With Managed Disks, encryption keys are managed by the IaaS, so you need not (and cannot) supply your own keys. You also do not need to re-create VMs after encrypting disks or rotating encryption keys, because the IaaS propagates the change to all VMs automatically.

Encrypt Azure Disks

Follow the steps below to initiate or rotate disk encryption for BOSH-deployed VMs on Azure:

1. Log in to Azure Portal and follow the Encryption workflow to encrypt new and existing PCF VMs.
2. For unmanaged Storage Account disks, follow the Recreate BOSH-Deployed Disks procedure below to propagate the change to existing VMs.
   * With Managed Disks, you can skip this step.

For more information about how BOSH integrates with IaaS-level disk encryption on Azure, see Encryption under Microsoft Azure in the BOSH documentation.

vSphere v6.5 or Later

vSphere supports disk encryption in versions 6.5 and later for encrypted VMs. Follow the steps below to initiate or rotate disk encryption for BOSH-deployed VMs on vSphere v6.5+:

1. Log in to vCenter and follow the Encrypt an Existing Virtual Machine or Virtual Disk procedure in the VMware Docs.
2. Follow the Recreate BOSH-Deployed Disks procedure below to propagate the change to existing VMs.

For more information about how BOSH integrates with IaaS-level disk encryption on vSphere, see Encryption under vSphere in the BOSH documentation.

AWS (Ops Manager v2.0 and later)

On AWS, you can use your Amazon account key to encrypt Linux EBS volumes, or supply your own key.

For instructions on how to encrypt BOSH-deployed VMs and the Ops Manager VM on AWS, see Configuring Amazon EBS Encryption.

For more information about how BOSH integrates with IaaS-level disk encryption on AWS, see Encryption under Amazon Web Services in the BOSH documentation.

Recreate BOSH-Deployed Disks

Unless you are using Azure Managed Disks, you need to manually recreate disks on BOSH-deployed VMs after you have added or rotated disk encryption keys. To manually recreate disks, do the following:

1. Configure Ops Manager to encrypt VM root, ephemeral disk, and persistent disk on next deploy:
   * Root File System: To recreate the root file system for VMs, you must upload a new stemcell. If you are already running the latest stemcell, you can:
     * Wait until a new stemcell comes out, typically less than two weeks.
     * Call Pivotal Support if propagating disk encryption is urgent.
   * Ephemeral Disks: In the Director Config pane of the Ops Manager tile, enable the Recreate All VMs checkbox.
   * Persistent Disks
     * PCF v2.3 and later: In the Director Config pane of the Ops Manager tile, enable the Recreate All Persistent Disks checkbox.
     * PCF v2.2 and earlier: In the Resource Config pane of all tiles, change the disk or VM sizes of all VMs that you need to encrypt.
2. Click Review Pending Changes and Apply Changes.
Security Event Logging

This topic describes how to enable and interpret security event logging for the Cloud Controller, the User Account and Authentication (UAA) server, and CredHub. Operators can use these logs to retrieve information about a subset of requests to the Cloud Controller, UAA server, and CredHub for security or compliance purposes.

Cloud Controller Logging

The Cloud Controller logs security events to syslog. You must configure a syslog drain to forward your system logs to a log management service.

See the Configuring System Logging in PAS topic for more information.

Format for Log Entries

Cloud Controller logs security events in the Common Event Format (CEF). CEF specifies the following format for log entries:

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension
```

Entries in the Cloud Controller log use the following format:

```
CEF:CEF_VERSION|cloud_foundry|cloud_controller_ng|CC_API_VERSION|SIGNATURE_ID|NAME|SEVERITY|rt=TIMESTAMP|user=USERNAME|suid=USER_GUID
cs1Label=userAuthenticationMechanism cs1=AUTH_MECHANISM
cs2Label=vcapRequestId cs2=VCAP_REQUEST_ID request=REQUEST requestMethod=REQUEST_METHOD
cs3Label=result cs3=RESULT
cs4Label=httpStatusCode cs4=HTTP_STATUS_CODE src=SOURCE_ADDRESS
dst=DESTINATION_ADDRESS
cs5Label=xForwardedFor cs5=X_FORWARDED_FOR_HEADER
```

Refer to the following list for a description of the properties shown in the Cloud Controller log format:

- **CEF_VERSION**: The version of CEF used in the logs.
- **CC_API_VERSION**: The current Cloud Controller API version.
- **SIGNATURE_ID**: The method and path of the request. For example, `/v2/app:GUID`.
- **NAME**: The same as SIGNATURE_ID.
- **SEVERITY**: An integer that reflects the importance of the event.
- **TIMESTAMP**: The number of milliseconds since the Unix epoch.
- **USERNAME**: The name of the user who originated the request.
- **USER_GUID**: The GUID of the user who originated the request.
- **AUTH_MECHANISM**: The user authentication mechanism. This can be `oauth-access-token`, `basic-auth`, or `no-auth`.
- **VCAP_REQUEST_ID**: The VCAP request ID of the request.
- **REQUEST**: The request path and parameters. For example, `/v2/info?MY-PARAM=VALUE`.
- **REQUEST_METHOD**: The method of the request. For example, `GET`.
- **RESULT**: The meaning of the HTTP status code of the response. For example, `success`.
- **HTTP_STATUS_CODE**: The HTTP status code of the response. For example, `200`.
- **SOURCE_ADDRESS**: The IP address of the client who originated the request.
- **DESTINATION_ADDRESS**: The IP address of the Cloud Controller VM.
- **X_FORWARDED_FOR_HEADER**: The contents of the X-Forwarded-For header of the request. This is empty if the header is not present.

Example Log Entries

The following list provides several example requests with the corresponding Cloud Controller log entries.
An anonymous GET request:

```plaintext
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET	/v2/info
/v2/info?tenant_id=1466090037402\nuser=\nsuid=\nrequest=\nrequestMethod=\nsrc=127.0.0.1\ndst=192.0.2.1
```

A GET request with basic authentication:

```plaintext
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET	/v2/syslog_drain_urls
/v2/syslog_drain_urls?batch_size=1000\nuser=bulk_api\nsuid=\nrequestMethod=\nsrc=127.0.0.1\ndst=192.0.2.1
```

A GET request with OAuth access token authentication:

```plaintext
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET	/v2/routes
/v2/routes?inline-relations-depth=1&q=host=dora&domain_guid=777-1o9f-5f5n-i888-o2025cb2dfc3
```

A GET request that results in a 404 error:

```plaintext
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET	/v2/apps/7f310103-39aa-4a8c-b92a-9ff8a6a2fa6b
```

A POST request that results in a 403 error:

```plaintext
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|POST	/v2/apps
```

---

**UAA Logging**

UAA logs security events to a file located at `/var/vcap/sys/log/uaa/uaa.log` on the UAA virtual machine (VM). Because these logs are automatically rotated, you must configure a syslog drain to forward your system logs to a log management service.

See the Configuring System Logging in PAS topic for more information.

**Log Events**

UAA logs identify the following categories of events:

- **Authorization and Password Events**
- **SCIM Administration Events**
• Token Events
• Client Administration Events
• UAA Administration Events

To learn more about the names of the events included in these categories and the information they record in the UAA logs, see User Account and Authentication Service Audit Requirements [7].

Example Log Entries

The following sections provide several example requests with the corresponding UAA log entries.

Successful User Authentication

Audit: TokenIssuedEvent ("openid","scim.read","uaa.user","cloud_controller.read","password.write","cloud_controller.write","scim.write"):
principal=a42026d6-5533-1884-eef2-838abcd85c3,
origin=[client=cf, user=bob], identityZoneId=[uaa]

This entry records a TokenIssuedEvent.

UAA issued a token associated with the scopes "openid","scim.read","uaa.user","cloud_controller.read","password.write","cloud_controller.write","scim.write" to the user bob.

Failed User Authentication

Audit: UserAuthenticationFailure (bob@example.com):
principal=61965469-c821-46b7-825f-630e12a51d6c,
origin=[remoteAddress=198.51.100.1, clientId=cf],
identityZoneId=[uaa]

This entry records a UserAuthenticationFailure.

The user bob@example.com originating at 198.51.100.1 failed to authenticate.

Successful User Creation

Audit: UserCreatedEvent ("user_id=61965469-c821-46b7-825f-630e12a51d6c","username=bob@example.com"):
principal=91220262-d901-44c0-825f-633i33b55d6c,
origin=[client=cf, user=admin, details=(198.51.100.1, tokenType=bearertokenValue=<TOKEN>, sub=20i03423-dd8e-33e1-938d-e9999e30f500, iss=https://uaa.example.com/oauth/token), identityZoneId=[uaa]

This entry records a UserCreatedEvent.

The admin user originating at 198.51.100.1 created a user named bob@example.com.

Successful User Deletion

Audit: UserDeletedEvent ("user_id=61965469-c821-46b7-825f-630e12a51d6c","username=bob@example.com"):
principal=61965469-c821-46b7-825f-630e12a51d6c,
origin=[client=admin, details=(remoteAddress=198.51.100.1, tokenType=bearertokenValue=<TOKEN>, sub=admin, iss=https://uaa.example.com/oauth/token), identityZoneId=[uaa]

This entry records a UserDeletedEvent.

The admin user originating at 198.51.100.1 deleted a user named bob@example.com.
CredHub Logging

CredHub logs security events to a file located at `/var/vcap/sys/log/credhub/credhub_security_events.log` on the CredHub VM. Because these logs are automatically rotated, you must configure a syslog drain to forward your system logs to a log management service.

See the Configuring System Logging in PAS topic for more information.

Format for Log Entries

CredHub logs security events in the Common Event Format (CEF). CEF specifies the following format for log entries:

CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension

Entries in the CredHub log use the following format:

CEF:0|cloud_foundry|credhub|CREDHUB_SERVER_VERSION|SIGNATURE_ID|NAME|0|rt=TIMESTAMP user=USERNAME suid=USER_GUID cs1Label=userAuthenticationMechanism cs1=AUTH_MECHANISM request=REQUEST requestMethod=REQUEST_METHOD cs3Label=versionUuid cs3=VERSION_UUID cs4Label=httpStatusCode cs4=HTTP_STATUS_CODE src=SOURCE_ADDRESS dst=DESTINATION_ADDRESS cs2Label=resourceName cs2=RESOURCE_NAME cs5Label=resourceUuid cs5=RESOURCE_UUID deviceAction=OPERATION cs6Label=requestDetails cs6=REQUEST_DETAILS

Refer to the following list for a description of the properties shown in the CredHub log.

- **CEF_VERSION**: The version of CEF used in the logs.
- **CREDHUB_SERVER_VERSION**: The current CredHub server version.
- **SIGNATURE_ID**: The method and path of the request. For example, `GET /v2/app:GUID`.
- **NAME**: The same as `SIGNATURE_ID`.
- **TIMESTAMP**: The number of milliseconds since the Unix epoch.
- **USERNAME**: The name of the user who originated the request, as defined by UAA. In the case of mTLS, no-auth, or not defined, this value is empty.
- **USER_GUID**: The “actor” identifier. For example, `mtls-app:GUID`. If there is no authenticated user, this value is empty.
- **AUTH_MECHANISM (cs1)**: The user authentication mechanism. This can be `oauth-access-token`, `mtls-auth`, or `no-auth`.
- **REQUEST**: The request path and parameters. For example, `/v2/info?MY-PARAM=VALUE`.
- **REQUEST_METHOD**: The method of the request. For example, `GET`.
- **HTTP_STATUS_CODE (cs4)**: The HTTP status code of the response. For example, `200`.
- **SOURCE_ADDRESS**: The IP address of the client who originated the request.
- **DESTINATION_ADDRESS**: The IP address of the CredHub VM.
- **RESOURCE_NAME (cs2)**: The credential path name. For example, `/my/awesome/credential`.
- **RESOURCE_UUID (cs5)**: The top-level credential UUID. This is not the credential version.
- **VERSION_UUID (cs3)**: The credential version UUID.
- **OPERATION (deviceAction)**: The device action. The possible operations include the following:
  - `get`
  - `set`
  - `generate`
  - `regenerate`
  - `bulk-regenerate`
  - `delete`
  - `find`
  - `get-permissions`
  - `add-permission`
  - `delete-permission`
  - `interpolate`
  - `info`
  - `version`
Example Log Entries

The following sections provide several example requests with the corresponding CredHub log entries.

Accessing a Credential

CEF:0|cloud_foundry|credhub|2.0.0-beta.28|GET /api/v1/data
GET /api/v1/data
rt=1530901816757
suser=app:3c538393-d192-4e23-8c83-456654b3fa6c
suid=mtls-app:3c538393-d192-4e23-8c83-456654b3fa6c
cs1Label=userAuthenticationMechanism
cs1=mutual_tls
request=/api/v1/data?path=0b353167-0d5a-48c7-5036-7eaa
requestMethod=GET

cs2Label=resourceName
cs2=null

cs3Label=versionUuid
cs3=null

cs4Label=httpStatusCode
cs4=200	src=10.0.0.1
dst=credhub.service.cf.internal

cs5Label=resourceUuid
cs5=null	deviceAction=FIND

cs6Label=requestDetails

cs6={"nameLike":null,"path":"0b353167-0d5a-48c7-5036-7eaa","paths":null}

- A user authenticated to the CredHub instance using **mutual_tls** from **10.0.0.1**.
- The authenticated user accessed a CredHub credential.

Setting a Credential

CEF:0|cloud_foundry|credhub|2.0.0-beta.28|PUT /api/v1/data
PUT /api/v1/data
rt=1530901097821
user:cc_service_key_client suser=uaa-client:cc_service_key_client
userAuthenticationMechanism
requestMethod=PUT

request=/api/v1/data	requestMethod=PUT
request=/api/v1/data

httpStatusCode
cs4=200	src=10.0.0.1
dst=credhub.service.cf.internal

resourceUuid
cs5=ccda25c5-a40a-4512-b6f5-a42f8c3b5c4c
deviceAction=SET

requestDetails

cs6={"name":"/1530901097842989110","type":"json","mode":null,
"additionalPermissions":
["actor":null,"allowedOperations":[]]}

- A user authenticated to the CredHub instance with UAA from **10.0.0.1**.
- The authenticated user set a CredHub credential.

Generating a Credential

Note: The CredHub logs include descriptions for each custom label. For example, the logs include `cs2Label=resourceName` to define the `cs2` label. The value for `resourceName` appears in the log next to `cs2=/path/to/credential`.

© Copyright Pivotal Software Inc, 2013-2019 1281 2.5
- A user authenticated to the CredHub instance using [mutual_tls] from [10.0.0.1].
- The authenticated user generated a password credential named [/my/awesome/credential].
Network Security
This section introduces some of the networking and routing security options for your Pivotal Cloud Foundry (PCF) deployment.

Securing Traffic and Controlling Routes
You can enable and configure a number of customization options to secure traffic in and out of your PCF deployment.

- TLS Connections in PCF Deployments
- Securing Traffic into Cloud Foundry
- Providing a Certificate for Your SSL/TLS Termination Point
- Enabling TCP Routing

Using the IPsec Add-On
The IPsec add-on for PCF provides additional security to the network layer for each BOSH-deployed virtual machine (VM).

The PCF IPsec add-on secures network traffic within a Cloud Foundry deployment and provides internal system protection if a malicious actor breaches your firewall.

- Securing Data in Transit with the IPsec Add-on
- Rotating IPsec Credentials
- Installing the Pivotal Cloud Foundry IPsec Add-On

Network Communication Paths in PCF

- BOSH DNS Network Communications
- Cloud Controller Network Communications
- Container-to-Container Network Communications
- CredHub Network Communications
- Diego Network Communications
- Loggregator Network Communications
- MySQL Network Communications
- NATS Network Communications
- Routing Network Communications
- UAA Network Communications
BOSH DNS Network Communications

This topic describes BOSH DNS internal network communication paths with other Pivotal Application Service (PAS) components.

**BOSH DNS Communications**

The following table lists network communication paths for BOSH DNS. For information about configuring BOSH DNS in the BOSH Director, see [BOSH DNS Config Pane in the BOSH Director Tile](#).

---

**Note:** Port 8853 is the destination port for communications between BOSH DNS health processes. You must allow TCP traffic on 8853 for all VMs running BOSH DNS.

---

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any VM running BOSH DNS</td>
<td>backup-prepare</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>ccdcb</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>clock_global</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>cloud_controller</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>cloud_controller_worker</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>consul_server</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>diego_brain</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>diego_cell</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>diego_database</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>doppler</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>ha_proxy</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>loggregator_trafficcontroller</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>mysql_proxy</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>mysql_monitor</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>nats</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>nfs_server</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>router</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>syslog_adapter</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>syslog_scheduler</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>tcp_router</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>tcp_router</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>uaa</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Applies only to deployments where internal MySQL is selected as the database.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
<td>-----</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>uaadb</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>Service tile VMs</td>
<td>53</td>
<td>TCP and UDP</td>
<td>DNS</td>
<td>Unencrypted. This communication happens inside the VM.</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>backup-prepare</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>ccdb</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>clock_global</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>cloud_controller</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>cloud_controller_worker</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>consul_server</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>diego_brain</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>diego_cell</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>diego_database</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>doppler</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>ha_proxy</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>loggregator_trafficcontroller</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>mysql_proxy</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>mysql_monitor</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>nats</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>nfs_server</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>router</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>syslog_adapter</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>syslog_scheduler</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>tcp_router</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>uaa</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>uaadb</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any VM running BOSH DNS</td>
<td>Service tile VMs</td>
<td>8853</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
</tbody>
</table>

* Applies only to deployments where internal MySQL is selected as the database.
† Applies only to deployments where the internal NFS server is selected for file storage.
Cloud Controller Network Communications

This topic describes Cloud Controller internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to the Cloud Controller.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller</td>
<td>cloud_controller (Routing API)</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_brain</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_brain (SSH Proxy)</td>
<td>cloud_controller</td>
<td>9022</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_database (BBS)</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>doppler (Syslog Drain Binder)</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>router</td>
<td>cloud_controller</td>
<td>9022</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
</tbody>
</table>

Outbound Communications

The following table lists network communication paths that are outbound from the Cloud Controller.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller</td>
<td>mysql_proxy†</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication††</td>
</tr>
<tr>
<td>cloud_controller</td>
<td>nfs_server or other blobstore†</td>
<td>4443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS and basic authentication</td>
</tr>
<tr>
<td>cloud_controller</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth 2.0 or none‡</td>
</tr>
<tr>
<td>cloud_controller</td>
<td>diego_database (BBS)‡</td>
<td>8889</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>cloud_controller (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>cloud_controller (Routing API)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>cloud_controller_worker</td>
<td>mysql_proxy†</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication††</td>
</tr>
<tr>
<td>cloud_controller_worker</td>
<td>nfs_server or other blobstore†</td>
<td>4443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS and basic authentication</td>
</tr>
<tr>
<td>clock_global</td>
<td>mysql_proxy†</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication††</td>
</tr>
</tbody>
</table>

* Applies only to deployments where internal MySQL is selected as the database.
† MySQL authentication uses the MySQL native password method.
‡ The destination depends on your file storage or blobstore configuration.
¶ The authentication method depends on the type of request.

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
Container-to-Container Networking Communications

This topic describes Container-to-Container Networking internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to Container-to-Container Networking.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>diego_cell (Silk CNI)</td>
<td>diego_cell (Silk Daemon)</td>
<td>23954</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>diego_cell (Silk Daemon)</td>
<td>diego_api (Silk Controller)</td>
<td>4103</td>
<td>TCP</td>
<td>HTTP</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (VXLAN Policy Agent)</td>
<td>diego_database (api - Policy Server Internal)</td>
<td>4003</td>
<td>TCP</td>
<td>HTTP</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (BOSH DNS Adapter)</td>
<td>diego_brain (Service Discovery Controller)</td>
<td>8054</td>
<td>TCP</td>
<td>HTTP</td>
<td>Mutual TLS</td>
</tr>
</tbody>
</table>

Outbound Communications

The following table lists network communication paths that are outbound from Container-to-Container Networking.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>diego_database (api - Policy Server)</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>diego_database (api - Policy Server)</td>
<td>cloud_controller (api - Cloud Controller)</td>
<td>9022</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_database (api - Policy Server)</td>
<td>mysql_proxy*</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>diego_brain (Service Discovery Controller)</td>
<td>nats (NATS)</td>
<td>4222</td>
<td>TCP</td>
<td>HTTP</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>diego_cell (BOSH DNS)</td>
<td>diego_cell (BOSH DNS Adapter)</td>
<td>8053</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>diego_cell (VXLAN Policy Agent)</td>
<td>mysql_proxy*</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
</tbody>
</table>

*Applies only to deployments where internal MySQL is selected as the database.

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
CredHub Network Communications

This topic describes CredHub internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to the CredHub.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller (api)</td>
<td>credhub</td>
<td>8844</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_cell</td>
<td>credhub</td>
<td>8844</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS†</td>
</tr>
<tr>
<td>windows_cell</td>
<td>credhub</td>
<td>8844</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS†</td>
</tr>
<tr>
<td>windows2016_cell</td>
<td>credhub</td>
<td>8844</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS†</td>
</tr>
</tbody>
</table>

Outbound Communications

The following table lists network communication paths that are outbound from the CredHub.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>credhub</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>n/a</td>
</tr>
<tr>
<td>credhub</td>
<td>mysql_proxy*</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication**</td>
</tr>
</tbody>
</table>

*Applies only to deployments where internal MySQL is selected as the database.

**MySQL authentication uses the MySQL native password method.

† Diego cells use the certificate pairs generated for individual containers to authenticate with CredHub on behalf of applications.

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
Diego Network Communications

This topic describes Diego internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to Diego.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller</td>
<td>diego_database (BBS)</td>
<td>8889</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>cloud_controller (Routing API)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
</tbody>
</table>

Diego Internal Communications

The following table lists network communication paths that are internal for Diego.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>diego_brain (Auctioneer)</td>
<td>diego_cell (Rep)</td>
<td>1801</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_brain (Auctioneer)</td>
<td>diego_database (BBS)</td>
<td>8889</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_brain (Auctioneer)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_brain (SSH Proxy)</td>
<td>diego_database (BBS)</td>
<td>8889</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_brain (SSH Proxy)</td>
<td>diego_cell (App instances)</td>
<td>Varies</td>
<td>TCP</td>
<td>SSH</td>
<td>SSH</td>
</tr>
<tr>
<td>diego_brain (TPS Watcher)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (local Route Emitter)</td>
<td>diego_database (BBS)</td>
<td>8889</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>diego_brain (CC Uploader)</td>
<td>9091</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>diego_brain (File Server)^‡</td>
<td>8080</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>diego_database (BBS)</td>
<td>8889</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_database (BBS)</td>
<td>diego_brain (Auctioneer)</td>
<td>9016</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_database (BBS)</td>
<td>diego_cell (Rep)</td>
<td>1801</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_database (BBS)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
</tbody>
</table>

Outbound Communications

The following table lists network communication paths that are outbound from Diego.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>diego_brain</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_brain (SSH Proxy)</td>
<td>cloud_controller</td>
<td>9022</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_brain (SSH Proxy)</td>
<td>uaa</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS and OAuth 2.0</td>
</tr>
<tr>
<td>diego_cell (local Route Emitter)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>diego_cell (Rep)</td>
<td>nfs_server or other blobstore</td>
<td>Varies</td>
<td>TCP</td>
<td>HTTP</td>
<td>Signed URLs/TLS</td>
</tr>
<tr>
<td>diego_database (BBS)</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
</tbody>
</table>
The destination depends on your PAS blobstore configuration. If you use the internal blobstore, the Diego Cell communicates to the blobstore using TLS on port 4443.

MySQL authentication uses the MySQL native password method.

The Diego File Server is responsible for distributing non-sensitive, static platform assets to internal platform components.

Applies only to deployments where internal MySQL is selected as the database.

These are the host-side ports that map to port 2222 in app instance containers and are typically within the range 61001 to 65534.

**BOSH DNS Communications**

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
Loggregator Network Communications

This topic describes Loggregator internal network communication paths with other Pivotal Application Service (PAS) components.

Loggregator Communications

The following table lists network communication paths for Loggregator.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any*</td>
<td>loggregator_trafficcontroller</td>
<td>8081</td>
<td>TCP</td>
<td>HTTP/WebSocket</td>
<td>OAuth</td>
</tr>
<tr>
<td>Any VM running Metron</td>
<td>doppler</td>
<td>8082</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller</td>
<td>doppler</td>
<td>8082</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller (Reverse Log Proxy)</td>
<td>doppler</td>
<td>8082</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any*</td>
<td>loggregator_trafficcontroller (cf auth proxy)</td>
<td>8083</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth</td>
</tr>
<tr>
<td>loggregator_trafficcontroller (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>loggregator_trafficcontroller (metrics_forwarder)</td>
<td>BOSH Director (metrics_server)</td>
<td>25595 and 8443</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller</td>
<td>loggregator_trafficcontroller (Reverse Log Proxy)</td>
<td>8082</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>syslog_adapter</td>
<td>loggregator_trafficcontroller (Reverse Log Proxy)</td>
<td>8080</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>syslog_adapter</td>
<td>Any**</td>
<td>Any***</td>
<td>TCP</td>
<td>TCP, TCP w/ TLS, HTTPS</td>
<td>Basic authentication****</td>
</tr>
<tr>
<td>syslog_scheduler</td>
<td>syslog_adapter</td>
<td>4443</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>syslog_scheduler</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller (Reverse Log Proxy Gateway)</td>
<td>cloud_controller</td>
<td>9023</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any*</td>
<td>loggregator_trafficcontroller (Reverse Log Proxy Gateway)</td>
<td>8088</td>
<td>TCP</td>
<td>HTTP/Server Sent Events</td>
<td>OAuth</td>
</tr>
</tbody>
</table>

*Any source VM can send requests to the specified destination within its subnet.
**Any host configured through a user-provided service binding with a syslog URL.
***Any port configured through a user-provided service binding with syslog URL.
****Basic authentication only supported for HTTPS syslog drains.

Log Cache Communications

The following table lists network communication paths for Log Cache.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>loggregator_trafficcontroller (Reverse Log Proxy)</td>
<td>log-cache (Nozzle)</td>
<td>8082</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>Any*</td>
<td>log-cache</td>
<td>8080</td>
<td>TCP</td>
<td>gRPC over HTTP/2</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>gorouter</td>
<td>log-cache (Auth)</td>
<td>8083</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth</td>
</tr>
</tbody>
</table>
BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.

<table>
<thead>
<tr>
<th>Service</th>
<th>Type</th>
<th>Port</th>
<th>Protocol</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>log-cache (Auth Proxy)</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
</tr>
<tr>
<td>log-cache (Auth Proxy)</td>
<td>cloud_controller</td>
<td>9024</td>
<td>TCP</td>
<td>HTTPS</td>
</tr>
</tbody>
</table>

*Any source VM can send requests to the specified destination within its subnet.*
MySQL Network Communications

This topic describes MySQL internal network communication paths with other Pivotal Application Service (PAS) components.

Note: These communications only apply to deployments where internal MySQL is selected as the PAS database.

Inbound Communications

The following table lists network communication paths that are inbound to MySQL VMs.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>cloud_controller_worker</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>clock_global</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>credhub</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>diego_cell (VXLAN Policy Agent)</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>diego_database (Policy Server)</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>diego_database (BBS)</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>diego_database (Locket)</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>uaa</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
</tbody>
</table>

(*) MySQL authentication uses the MySQL native password method.

Internal Communications

The following table lists network communication paths that are internal to MySQL VMs.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql</td>
<td>mysql (Galera)</td>
<td>4567</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>mysql_monitor</td>
<td>mysql (MySQL Server)</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>mysql_monitor</td>
<td>mysql_proxy (Proxy health check)</td>
<td>443/8080**</td>
<td>TCP</td>
<td>MySQL</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>mysql_proxy</td>
<td>mysql (MySQL Server)</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>mysql_proxy</td>
<td>mysql (Galera health check)</td>
<td>9200</td>
<td>TCP</td>
<td>HTTP</td>
<td>Basic authentication</td>
</tr>
</tbody>
</table>

(*) MySQL authentication uses the MySQL native password method.

(**) Port 443 is used if mysql_proxy is registered with Gorouter. If not registered, mysql_proxy uses port 8080 instead.

Outbound Communications

The following table lists network communication paths that are outbound from MySQL.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql_monitor</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth</td>
</tr>
<tr>
<td>mysql_proxy (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
</tbody>
</table>

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
NATS Network Communications

This topic describes NATS internal network communication paths with other Pivotal Application Service (PAS) components.

Publish Communications

The following table lists network communications that are published to NATS.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>loggregator_trafficcontroller (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>mysql_proxy (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>nfs_server (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>uaa (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>diego_cell (local Route Emitter)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
</tbody>
</table>

*Applies only to deployments where internal MySQL is selected as the database.
†Applies only to deployments where the internal NFS server is selected for file storage.

Subscribe Communications

The following table lists network communications that are subscribed to NATS.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>router</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
</tbody>
</table>

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
Routing Network Communications

This topic describes the internal network communication paths of the routing subsystem with other Pivotal Application Service (PAS) components.

HTTP Routing

The following table lists network communication paths for HTTP routing.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>diego_cell (local Route Emitter)</td>
<td>nats</td>
<td>4222</td>
<td>TCPs</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>Load balancer</td>
<td>router (Gorouter)</td>
<td>80</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>Load balancer</td>
<td>router (Gorouter)</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>router (Gorouter)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
<tr>
<td>router (Gorouter)</td>
<td>System components</td>
<td>Varies</td>
<td>TCP</td>
<td>Varies</td>
<td>None</td>
</tr>
<tr>
<td>router (Gorouter)</td>
<td>App containers</td>
<td>Varies</td>
<td>TCP</td>
<td>Varies</td>
<td>Optional TLS</td>
</tr>
<tr>
<td>haproxy</td>
<td>router (Gorouter)</td>
<td>80</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>haproxy</td>
<td>router (Gorouter)</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>Load balancer</td>
<td>haproxy</td>
<td>80</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>Load balancer</td>
<td>haproxy</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
</tbody>
</table>

TCP Routing (Optional)

The following table lists network communication paths for TCP routing.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller</td>
<td>cloud_controller (Routing API)</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS and OAuth 2.0</td>
</tr>
<tr>
<td>cloud_controller (Routing API)</td>
<td>diego_database (Locket)</td>
<td>8891</td>
<td>TCP</td>
<td>HTTPS</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>cloud_controller (Routing API)</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication**</td>
</tr>
<tr>
<td>cloud_controller (Routing API)</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>diego_brain (global TCP Emitter)</td>
<td>cloud_controller (Routing API)</td>
<td>3000</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_brain (global TCP Emitter)</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>diego_cell (local Route Emitter)</td>
<td>cloud_controller (Routing API)</td>
<td>3000</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>diego_cell (local Route Emitter)</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>Load balancer</td>
<td>tcp_router</td>
<td>1024-65535</td>
<td>TCP</td>
<td>TCP</td>
<td>None</td>
</tr>
<tr>
<td>router (Gorouter)</td>
<td>cloud_controller (Routing API)</td>
<td>3000</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>router (Gorouter)</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>tcp_router</td>
<td>cloud_controller (Routing API)</td>
<td>3000</td>
<td>TCP</td>
<td>HTTP</td>
<td>OAuth 2.0</td>
</tr>
<tr>
<td>tcp_router</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
</tbody>
</table>

* This communication happens through a load balancer and a Gorouter. Requests are received by Routing API on port 3000.
You can use this port range to configure the port in the PAS tile.

MySQL authentication uses the MySQL native password method.

Service Mesh (Optional)

The following table lists network communication paths for service mesh.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller (cloud_controller_ng)</td>
<td>istio_control (copilot)</td>
<td>9001</td>
<td>TCP</td>
<td>GRPC</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>istio_control (copilot)</td>
<td>diego_database (bbs)</td>
<td>8889</td>
<td>TCP</td>
<td>HTTP</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>istio_control (pilot-discovery)</td>
<td>istio_control (copilot)</td>
<td>9009</td>
<td>TCP</td>
<td>GRPC</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>istio_router (envoy)</td>
<td>App Containers</td>
<td>Varies</td>
<td>TCP</td>
<td>HTTP/HTTPS</td>
<td>Optional TLS</td>
</tr>
<tr>
<td>istio_router (envoy)</td>
<td>istio_control (pilot-discovery)</td>
<td>15010</td>
<td>TCP</td>
<td>GRPC</td>
<td>None</td>
</tr>
<tr>
<td>Load balancer</td>
<td>istio_router (envoy)</td>
<td>80</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>Load balancer</td>
<td>istio_router (envoy)</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>Load balancer (health_check)</td>
<td>istio_router (envoy)</td>
<td>8002</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>route_syncer (cc_route_syncer)</td>
<td>istio_control (copilot)</td>
<td>9001</td>
<td>TCP</td>
<td>GRPC</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>route_syncer (cc_route_syncer)</td>
<td>mysql_proxy*</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication*</td>
</tr>
<tr>
<td>n/a (admin)</td>
<td>istio_router (envoy)</td>
<td>8001</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>n/a (for envoy secure GRPC</td>
<td>istio_control (pilot-discovery)</td>
<td>15012</td>
<td>TCP</td>
<td>GRPC</td>
<td>Mutual TLS</td>
</tr>
<tr>
<td>communication)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n/a (for HTTP discovery service)</td>
<td>istio_control (pilot-discovery)</td>
<td>8080</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
<tr>
<td>n/a (for pilot’s self-monitoring)</td>
<td>istio_control (pilot-discovery)</td>
<td>9093</td>
<td>TCP</td>
<td>HTTP</td>
<td>None</td>
</tr>
</tbody>
</table>

* Applies only to deployments where internal MySQL is selected as the database.

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director colocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
UAA Network Communications

This topic describes User Account and Authentication (UAA) internal network communication paths with other Pivotal Application Service (PAS) components.

Inbound Communications

The following table lists network communication paths that are inbound to UAA.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth 2.0 or none</td>
</tr>
<tr>
<td>diego_brain (SSH Proxy)</td>
<td>uaa</td>
<td>443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>loggregator_trafficcontroller</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>TLS</td>
</tr>
<tr>
<td>mysql_monitor</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth</td>
</tr>
<tr>
<td>router</td>
<td>uaa</td>
<td>8443</td>
<td>TCP</td>
<td>HTTPS</td>
<td>OAuth 2.0</td>
</tr>
</tbody>
</table>

*The authentication method depends on the type of request.

Outbound Communications: Internal to PCF

The following table lists network communication paths that are outbound from UAA.

<table>
<thead>
<tr>
<th>Source VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Security and Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>uaa</td>
<td>mysql_proxy</td>
<td>3306</td>
<td>TCP</td>
<td>MySQL</td>
<td>MySQL authentication</td>
</tr>
<tr>
<td>uaa (Route Registrar)</td>
<td>nats</td>
<td>4222</td>
<td>TCP</td>
<td>NATS</td>
<td>Basic authentication</td>
</tr>
</tbody>
</table>

* Applies only to deployments where internal MySQL is selected as the database.

** MySQL authentication uses the MySQL native password method.

Outbound Communications: External to PCF

The following table lists network communication paths from UAA that are outbound to external systems.

<table>
<thead>
<tr>
<th>Source VM VM</th>
<th>Destination VM</th>
<th>Port</th>
<th>Transport Layer Protocol</th>
<th>App Layer Protocol</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>uaa</td>
<td>LDAP</td>
<td></td>
<td>TCP</td>
<td>LDAP/LDAPS</td>
<td>Basic authentication (LDAP bind)</td>
</tr>
<tr>
<td>uaa</td>
<td>SAML/OIDC</td>
<td>80 or 443 (HTTP port)</td>
<td>TCP</td>
<td>HTTP/HTTPS</td>
<td>Key</td>
</tr>
</tbody>
</table>

BOSH DNS Communications

By default, PAS components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director collocates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS Network Communications.
Credential and Identity Management

This section provides links to different aspects of identity management, including credential management handled by CredHub, user creation and permissions management, and authentication for Pivotal Cloud Foundry (PCF).

General Identity Management

The following topics provide general information about credential and identity management in PCF.

- Pivotal Cloud Foundry User Types
- Retrieving Credentials from Your Deployment

CredHub Documentation

CredHub provides centralized credential management in Pivotal Cloud Foundry (PCF). Credentials can include passwords, certificates, and SSH keys.

CredHub centralizes and secures credential generation, storage, lifecycle management, and system access.

For more information about CredHub, see the following topics.

- CredHub: Provides an overview of CredHub.
- CredHub Credential Types: Provides a reference of credential types supported in CredHub.

UAA Documentation

PCF uses UAA to manage account roles and permissions in PCF runtimes. UAA supports access control as OAuth2 services and can store user information internally, or connect to external user stores through LDAP or SAML.

For more information about UAA, see the following topics.

- UAA Overview
- UAA Concepts
- UAA Architecture
- Identity Providers in UAA

PCF Roles and User Accounts

The following topics describe how to manage PCF roles and user accounts in PCF.

- Creating and Managing Users with the UAA CLI (UAAC)
- Adding Existing SAML or LDAP Users to a PCF Deployment

Ops Manager Roles and Permissions

- Creating UAA Clients for BOSH Director
- Configuring Role-Based Access Control (RBAC) in Ops Manager
- Creating and Managing Ops Manager User Accounts

PAS Roles and Permissions

This section provides links to Pivotal Application Service (PAS) roles and permissions documentation.
PKS Roles and Permissions

This section provides links to Pivotal Container Service (PKS) roles and permissions documentation.

- Managing Users in PKS with UAA

Pivotal Cloud Foundry User Types

This topic describes the types of users in a Pivotal Cloud Foundry (PCF) deployment, their roles and permissions, and who creates and manages their user accounts.

The users who run a PCF deployment and have admin privileges are operators. With Pivotal Application Service (PAS) installed to host apps, you add two more user types: PAS users who develop the apps and manage the development environment, and end users who just run the apps.

PCF distinguishes between these three user types and multiple user roles that exist within a single user type. Roles are assigned categories that more specifically define functions that a user can perform. A user may serve in more than one role at the same time.

Operators

Operators have the highest, admin-level permissions. We also refer to operators as Ops Manager admins and PAS admins because they perform an admin role within these contexts.

Tools and Tasks

Operators fulfill system administrator roles covering the entire PCF deployment. They work primarily with their IaaS and Ops Manager, to configure and maintain PAS component VMs. The component VMs, in turn, support the VMs that host applications. Typical operator tasks include:

- Deploying and configuring Ops Manager, PAS, and other product and service tiles.
- Maintaining and upgrading PCF deployments.
- Creating user accounts for PAS users and the orgs that PAS users work within.
- Creating service plans that define the access granted to end users.

User Accounts

When Ops Manager starts up for the first time, the operator specifies one of the following authentication systems for operator user accounts:

- Internal authentication, using a new UAA database that Ops Manager creates.
- External authentication, through an existing identity provider accessed through SAML protocol.

The operator can then use the UAAC to create more operator accounts.

PAS Users

PAS users are app developers, managers, and auditors who work within orgs and spaces, the virtual compartments within a deployment where PAS users can run apps and locally manage their roles and permissions.

A Role-Based Access Control (RBAC) system defines and maintains the different PAS user roles:

- Org Manager, Org Auditor, Org Billing Manager
- Space Manager, Space Developer, Space Auditor

The Orgs, Roles, Spaces, Permissions topic describes the PAS user roles, and what actions they can take within the orgs and spaces they belong to. Some of these permissions depend on the values of feature flags.

Tools

Space Developer users work with their software development tools and the apps deployed on host VMs.

All PAS users use system tools such as the Cloud Foundry Command Line Interface (cf CLI), PCF Metrics, and Apps Manager, a dashboard for managing PAS users, orgs, spaces, and apps.
User Accounts

When an operator configures PAS for the first time, they specify one of the following authentication systems for PAS user accounts:

1. Internal authentication, using a new UAA database created for PAS. This system-wide UAA differs from the Ops Manager internal UAA, which only stores Ops Manager Admin accounts.

2. External authentication, through an existing identity provider accessed through SAML or LDAP protocol.

In either case, PAS user role settings are saved internally in the Cloud Controller Database, separate from the internal or external user store.

Org and Space Managers then use Apps Manager to invite and manage additional PAS users within their orgs and spaces. PAS users with proper permissions can also use the cf CLI to assign user roles.

Operators can log into Apps Manager by using the **UAA Administrator User** credentials under the **Credentials** tab of the PAS tile. These UAA Admin credentials grant them the role of Org Manager within all orgs in the deployment. The UAA Admin can also use the UAAC to create new user accounts and the cf CLI to assign user roles.

End Users

End users are the people who log into and use the apps hosted on PAS. They do not interact directly with PAS components or interfaces. Any interactions or roles they perform within the apps are defined by the apps themselves, not Pivotal Application Service (PAS).

User Accounts and SSO

App developers can configure apps any way they want to grant end user access individually. In a deployment with Single Sign-On Service for Pivotal Cloud Foundry installed, they can also offer end users a single login that accesses multiple apps.

The Single Sign-On (SSO) service can save user account information in an external database accessed through SAML or LDAP, or in the internal PAS user store, along with PAS User accounts.

To make the SSO service available to developers, an operator creates service plans that give login access to specific groups of end users. A Space Manager then creates a local instance of the service plan, and registers apps with it. Apps registered to the plan instance then become available through SSO to all end users covered by the plan.

User Types Summary

The following table summarizes PCF user types, their roles, the tools they use, the System of Record (SOR) that stores their accounts, and what accounts they can provision.

<table>
<thead>
<tr>
<th>User Type</th>
<th>Available Roles</th>
<th>Tools They Use</th>
<th>Account SOR</th>
<th>Accounts They Can Provision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operators</td>
<td>Admin (UAA Admin, SSO Plan Admin, other system admins)</td>
<td>- IaaS UI</td>
<td>Ops Manager user store through UAA or External store through SAML</td>
<td>Operators and PAS Users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- PivNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Ops Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- cf CLI</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- UAA CLI (UAAC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SSO Dashboard</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Marketplace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAS Users</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- UAA Administrator</td>
<td>- cf CLI</td>
<td>PAS user store through UAA or External store through SAML</td>
<td>PAS Users within permitted orgs and spaces, and End Users</td>
</tr>
<tr>
<td></td>
<td>- Org Manager</td>
<td>- PivNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Org Auditor</td>
<td>- Ops Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Org Billing Manager</td>
<td>- cf CLI</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Space Manager</td>
<td>- PivNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Space Developer</td>
<td>- Apps Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Space Auditor</td>
<td>- PCF Metrics</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Marketplace</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
| End Users | Defined by apps they use | Hosted apps | Individual apps or PAS user store through SSO |
Retrieving Credentials from Your Deployment

This topic describes how the credentials for your Pivotal Cloud Foundry (PCF) deployment are stored and how you can access them.

- What credentials does PCF store?
  - Many PCF components use credentials to authenticate connections, and PCF installations often have hundreds of active credentials. This includes certificates, virtual machine (VM) credentials, and credentials for jobs running on the VMs.

- Where does PCF store these credentials?
  - PCF stores credentials in either the Ops Manager database or BOSH CredHub. In PCF v1.11 and later, the BOSH Director VM includes a co-located CredHub instance. Ops Manager, Pivotal Application Service (PAS), and service tiles running on PCF can use this CredHub instance to store their credentials. For example, in PCF v1.12, PAS began migrating its credentials to CredHub. See the PAS Release Notes for a full list.

- When do I need to access these credentials?
  - You may need to access credentials for Ops Manager, PAS, and service tiles as part of regular administrative tasks in PCF, including troubleshooting. Many procedures in this documentation require you to retrieve credentials.

- How can I retrieve credentials?
  - The workflow for retrieving credentials depends on where they are stored. See the procedures below.

Retrieve Credentials Stored in BOSH CredHub

To retrieve credentials from CredHub using the Ops Manager API, do the following:

1. Perform the procedures in the Using the Ops Manager API topic to authenticate and access the Ops Manager API.

2. Use the Ops Manager API to retrieve a list of deployed products:

   ```
   $ curl https://OPS-MAN-FQDN/api/v0/deployed/products
   -X GET
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   
   Replace UAA-ACCESS-TOKEN with the access token recorded in the previous step.
   ```

3. In the response to the above request, locate the guid for the product from which you want to retrieve credentials. For example, if you want to retrieve PAS credentials, find the installation_name starting with cf- and copy its guid.

4. Run the following curl command to list the names of the credentials stored in CredHub for the product you selected. If you already know the name of the credential, you can skip this step.

   ```
   $ curl https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/variables
   -X GET
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   
   Replace PRODUCT-GUID with the value of guid from the previous step.
   ```

5. Run the following curl command to view the credential:

   ```
   $ curl https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/variables?name=VARIABLE-NAME
   -X GET
   -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   
   Replace VARIABLE-NAME with the name of the credential you want to retrieve.
   ```

Retrieve Credentials Stored in the Ops Manager Database

To retrieve credentials stored in the Ops Manager database and not CredHub, use the Ops Manager UI or API as outlined in the procedures below.
Retrieve Credentials Using the Ops Manager UI

1. From Ops Manager, select the product tile for which you want to retrieve credentials.
2. Click the **Credentials** tab.
3. Locate the credential that you need and click **Link to Credential**.

Retrieve Credentials Using the Ops Manager API

1. Perform the procedures in the Using the Ops Manager API topic to authenticate and access the Ops Manager API.
2. Use the Ops Manager API to retrieve a list of deployed products:

   ```bash
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products" \
     -X GET \
     -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ``
   Replace **UAA-ACCESS-TOKEN** with the access token recorded in the previous step.

3. In the response to the above request, locate the `guid` for the product from which you want to retrieve credentials. For example, if you want to retrieve PAS credentials, find the `installation_name` starting with `cf-` and copy its `guid`.

4. Run the following `curl` command to list references for the credentials stored in Ops Manager for the product you selected. If you already know the reference for the credential, you can skip this step.

   ```bash
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/credentials" \
     -X GET \
     -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ``
   Replace **PRODUCT-GUID** with the value of `guid` from the previous step.

5. Run the following `curl` command to view the credential:

   ```bash
   $ curl "https://OPS-MAN-FQDN/api/v0/deployed/products/PRODUCT-GUID/credentials/CREDENTIAL-REFERENCE" \
     -X GET \
     -H "Authorization: Bearer UAA-ACCESS-TOKEN"
   ``
   Replace **CREDENTIAL-REFERENCE** with the name of the credential you want to retrieve.

Changing Ops Manager Credentials

Ops Manager Password

1. Log in to Ops Manager and navigate to **My Account**. You can access this at: [https://OPS-MAN-FQDN/uaa/profile](https://OPS-MAN-FQDN/uaa/profile).

2. Navigate to **Change Password**. You can access this at: [https://OPS-MAN-FQDN/uaa/change_password](https://OPS-MAN-FQDN/uaa/change_password).
3. Enter your current password and a new password.
Ops Manager Decryption Passphrase

You must have the existing passphrase to update the decryption passphrase.

1. Log in to Ops Manager, and navigate to Settings. You can access this at https://OPS-MAN-FQDN/encryption_passphrase/edit.

2. In the Decryption Passphrase panel, enter your current decryption passphrase and the new decryption passphrase, then click Save.
S3 Compatible Blobstore Credentials

If you use an S3 compatible blobstore, you can rotate your blobstore credentials from the Director Config panel of the BOSH Director tile. After entering new credentials and clicking Review Pending Changes, then Apply Changes, BOSH recreates the VMs in your deployment to apply the new credentials.

1. From a browser, navigate to https://OPS-MAN-FQDN/ and log in to Ops Manager.

2. From the Installation Dashboard, click the BOSH Director tile.

3. From the Director Config panel, select the Recreate all VMs checkbox.
For **Access Key**, enter a new access key.

This will force BOSH to recreate all VMs on the next deploy. Persistent disk will be preserved.
5. For **Secret Key**, enter a new secret key.

6. Click **Review Pending Changes**, then **Apply Changes**.

7. Clear the **Recreate all VMs** checkbox.
CredHub

Overview

CredHub is a component designed for centralized credential management in Pivotal Cloud Foundry (PCF). It is a single component that can address several scenarios in the PCF ecosystem. At the highest level, CredHub centralizes and secures credential generation, storage, lifecycle management, and access.

What Can CredHub Do?

CredHub performs a number of different functions to help generate and protect the credentials in your PCF deployment.

- Securing data for storage
- Authentication
- Authorization
- Access and change logging
- Data typing
- Credential generation
- Credential metadata
- Credential versioning

Application Architecture

CredHub consists of a REST API and a CLI. The REST API conforms to the Config Server API spec. CredHub is an OAuth2 resource server that integrates with User Account Authentication (UAA) to provide core authentication and federation capabilities.

CredHub in PCF

A PCF deployment stores credentials in the following locations:

- **BOSH CredHub**: Colocated with the BOSH Director on a single VM. This CredHub instance stores credentials for the BOSH Director.
- **Runtime CredHub**: Deployed as an independent service and stores service instance credentials.

BOSH CredHub

In PCF, BOSH Director VM includes a CredHub job. This provides a lightweight credential storage instance for the BOSH Director. The BOSH Director, Pivotal Application Service (PAS), and other tiles store credentials in BOSH CredHub. For more information, see [Retrieving Credentials from Your Deployment](#).

Note: This configuration does not provide high availability.
In this collocated deployment architecture, the BOSH Director, CredHub, UAA, and the Director database are all installed on a single BOSH VM, as shown in the following diagram:

Runtime CredHub

The PAS tile deploys CredHub as an independent service on its own VM. This provides a highly available credential storage instance for securing service instance credentials. For more information, see Securing Service Instance Credentials with Runtime CredHub.

CredHub is a stateless application, so you can scale it to multiple instances that share a common database cluster and encryption provider.

With CredHub as a service, the load balancer and external databases communicate directly with the CredHub VMs, as shown in the following diagram:

Using CredHub to Store Credentials for Service Tiles

If you develop a service tile for PCF and want to store its credentials in BOSH CredHub, see the CredHub section of the Tile Developer Guide.

CredHub Credential Types

Credentials exist in multiple places in the PCF ecosystem. PCF components use credentials to authenticate connections between components. PCF installations often have hundreds of active credentials. Leaked credentials are common causes of data and security breaches, so managing them securely is very important.

For more information, read CredHub Credential Types.

Backing Up and Restoring CredHub Instances

The CredHub application does not hold state, but you must ensure its dependent components are backed up. Redundant backups can help prevent data loss if an individual component fails. For more information, read Backing Up and Restoring CredHub Instances.
CredHub Credential Types

This topic describes the different credential types supported by CredHub.

CredHub supports different types of credentials to simplify generating and managing multi-part credentials. For example, a TLS certificate contains three parts: the root certificate authority (CA), the certificate, and the private key. CredHub supports all three parts, which helps keep connection requests from being rejected erroneously.

CredHub supports the following credential types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A single string value for arbitrary configurations and other non-generated or validated strings.</td>
</tr>
<tr>
<td>json</td>
<td>An arbitrary JSON object for static configurations with many values.</td>
</tr>
<tr>
<td>user</td>
<td>Three string values for username, password, and password hash.</td>
</tr>
<tr>
<td>password</td>
<td>A single string value for passwords and other random string credentials. Values for this type can be automatically generated.</td>
</tr>
<tr>
<td>certificate</td>
<td>An object containing a root CA, certificate, and private key. Use this type for key pair applications that utilize a certificate, such as TLS connections. Values for this type can be automatically generated.</td>
</tr>
<tr>
<td>rsas</td>
<td>An object containing an RSA public key and private key without a certificate. Values for this type can be automatically generated.</td>
</tr>
<tr>
<td>ssh</td>
<td>An object containing an SSH-formatted public key and private key. Values for this type can be automatically generated.</td>
</tr>
</tbody>
</table>

Each credential type supports distinct parameters for customizing how credentials are generated. These include minimum password lengths, required characters, and certificate fields. For more information, see the Generate Credentials section of the CredHub API documentation.

For every credential type, secret values are encrypted before storage. For instance, the private key of a certificate-type credential and the password of a user-type credential are encrypted before storage. For JSON and Value type credentials, the full contents are encrypted before storage.
Security for Apps and Services

This topic links to topics that describe how PCF and PCF users manage security for apps and service instances.

- **Application Security Groups**: Describes how ASGs work and how to manage them in all versions of Cloud Foundry, including PCF.
- **Configuring SSH Access for PCF**: Explains how to configure PCF to allow SSH access to app instances, for debugging.
- **Restricting App Access to Internal PCF Components**: Details how to create Application Security Groups (ASGs), rules that allow internal outgoing communications from all apps in PCF, or the apps running in the same space.
- **Configuring Application Security Groups for Email Notifications**: Describes how to define an ASG to enable app-generated notifications.
- **Trusted System Certificates**: Explains where applications can find trusted system certificates.
- **Managing Access to Service Plans**: Describes how to enable or disable access to service plans for a subset of users.
- **Delivering Service Credentials to an Application**: Provides documentation on how to bind apps to service instances, which generate the credentials that enable the apps to use the service.
- **Managing Service Keys**: Explains to create and manage service keys that enable apps to use service instances.
Compliance and Other Security-Related Topics

This section contains topics related to compliance and securing Pivotal Cloud Foundry (PCF) deployments.

Compliance

- **NIST Controls and PCF**: Provides a dedicated site that assesses Pivotal Cloud Foundry against NIST SP 800-53(r4) Controls.
- **General Data Protection Regulation**: Provides an overview of the General Data Protection Regulation (GDPR) and where Pivotal Cloud Foundry (PCF) may store personal data

Other Security-Related Topics

- **Security-Related PCF Tiles and Add-ons**: Provides links to other security-related services available for PCF deployments.
- **Security Guidelines for Your IaaS Provider**: Provides links to security-related documentation related to different IaaS providers. The topics may come from third party providers, and are not necessarily owned or maintained by Pivotal. Pivotal provides them for your convenience, but cannot guarantee the accuracy or currency of these documents.
General Data Protection Regulation

This topic provides an overview of the General Data Protection Regulation (GDPR) and where Pivotal Cloud Foundry (PCF) may store personal data.

Overview

GDPR came into effect on May 25, 2018 and impacts any company processing the data of EU citizens or residents, even if the company is not EU-based. The GDPR sets forth how companies should handle privacy issues, securely store data, and respond to security breaches.

Understand Personal Data

The GDPR grants data subjects certain rights, such as the right to obtain a copy of their personal data, object to the processing of personal data, and the right to have their personal data erased. Organizations subject to GDPR need to ensure that they can address and respond to requests by data subjects if they are processing their personal data.

Article 4, Section 1 of the GDPR defines personal data as follows:

‘personal data’ means any information relating to an identified or identifiable natural person (‘data subject’); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;

For more information, see the GDPR text.

Personal data can be collected, stored, and processed in a PCF deployment. Pivotal has performed a review of PCF components and determined that personal data may reside in the following areas:

- User Account and Authentication (UAA)
- Cloud Foundry API
- Routing
- Diego
- Notifications Service

Where Personal Data May Reside

The following sections explain how different PCF components collect personal data.

User Account and Authentication (UAA)

UAA is an open-source Cloud Foundry component that provides identity management features and identity-based security for applications and APIs. For more information, see User Account and Authentication.

<table>
<thead>
<tr>
<th>GDPR</th>
<th>Workflow</th>
<th>What personal data is collected?</th>
<th>When is it collected?</th>
<th>Where is it stored?</th>
<th>How is it processed?</th>
<th>Who has access to it?</th>
</tr>
</thead>
</table>
|       | User registers | • Username  
• Email address  
• First name (optional)  
• Last name (optional)  
• User ID (UAA GUID, generated) | User registration submission | UAA DB | Stored in UAA DB | • End user  
• UAA administrators |
### Business Initiation

<table>
<thead>
<tr>
<th>Just-in-time provisioning: create user on user login</th>
</tr>
</thead>
<tbody>
<tr>
<td>• First name (optional)</td>
</tr>
<tr>
<td>• Last name (optional)</td>
</tr>
<tr>
<td>• User ID (UAA GUID, generated)</td>
</tr>
<tr>
<td>• Additional attributes as defined by the organization</td>
</tr>
<tr>
<td>User login</td>
</tr>
<tr>
<td>UAA DB</td>
</tr>
<tr>
<td>Stored in UAA DB</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Admin user makes a creation API call</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Username</td>
</tr>
<tr>
<td>• Email address</td>
</tr>
<tr>
<td>• First name (optional)</td>
</tr>
<tr>
<td>• Last name (optional)</td>
</tr>
<tr>
<td>• User ID (UAA GUID, generated)</td>
</tr>
<tr>
<td>• Additional attributes as defined by the organization</td>
</tr>
<tr>
<td>Admin API call</td>
</tr>
<tr>
<td>UAA DB</td>
</tr>
<tr>
<td>Stored in UAA DB</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

### Business Execution

<table>
<thead>
<tr>
<th>User self-updates profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Email address</td>
</tr>
<tr>
<td>• First name (optional)</td>
</tr>
<tr>
<td>• Last name (optional)</td>
</tr>
<tr>
<td>User registration submission</td>
</tr>
<tr>
<td>UAA DB</td>
</tr>
<tr>
<td>Stored in UAA DB</td>
</tr>
<tr>
<td>End user</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Just-in-time provisioning: user update</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Email address</td>
</tr>
<tr>
<td>• First name (optional)</td>
</tr>
<tr>
<td>• Last Name (optional)</td>
</tr>
<tr>
<td>• Additional attributes as defined by the organization</td>
</tr>
<tr>
<td>User login</td>
</tr>
<tr>
<td>UAA DB</td>
</tr>
<tr>
<td>Stored in UAA DB</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User logs in</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Current account cookie (generated)</td>
</tr>
<tr>
<td>• Saved account cookie (generated)</td>
</tr>
<tr>
<td>User login</td>
</tr>
<tr>
<td>User browser</td>
</tr>
<tr>
<td>By UAA</td>
</tr>
<tr>
<td>• End user</td>
</tr>
<tr>
<td>• UAA login page</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Admin user makes an update API call</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Email address</td>
</tr>
<tr>
<td>• First name (optional)</td>
</tr>
<tr>
<td>• Last name (optional)</td>
</tr>
<tr>
<td>• Additional attributes as defined by the organization</td>
</tr>
<tr>
<td>Admin API call</td>
</tr>
<tr>
<td>UAA DB</td>
</tr>
<tr>
<td>Stored in UAA DB</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

### Delete User Flow

<table>
<thead>
<tr>
<th>Admin user makes a hard delete API call</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
</tr>
<tr>
<td>Deleted from UAA DB</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Admin user makes a deactivation API call</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
</tr>
<tr>
<td>Soft delete (records still held in database but user cannot login)</td>
</tr>
<tr>
<td>UAA administrators</td>
</tr>
</tbody>
</table>

### Reports/Logs

<table>
<thead>
<tr>
<th>Event or debug logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any information</td>
</tr>
<tr>
<td>When event happens</td>
</tr>
<tr>
<td>UAA logs</td>
</tr>
<tr>
<td>Depends on setup of Loggregator and log forwarding</td>
</tr>
<tr>
<td>BOSH administrators</td>
</tr>
</tbody>
</table>

### Cloud Foundry API

The Cloud Foundry API release contains several components, including the Cloud Controller. For more information, see the Cloud Foundry API release README.
### GDPR Workflow

<table>
<thead>
<tr>
<th>What personal data is collected</th>
<th>When is it collected?</th>
<th>Where is it stored?</th>
<th>How is it processed?</th>
<th>Who has access to it?</th>
<th>How long is it kept?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Initiation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User makes a request for the first time</td>
<td>The first time a user makes a request to the API</td>
<td>Cloud Controller DB</td>
<td>It is used to identify permissions for the user</td>
<td>PCF operator</td>
<td>As long as the user is part of the system</td>
</tr>
<tr>
<td><strong>Business Execution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Troubleshooting API requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• User ID • User agent • IP address</td>
<td>On each request</td>
<td></td>
<td></td>
<td>PCF operator</td>
<td></td>
</tr>
<tr>
<td><strong>Audit Trails</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audit what changes a user makes</td>
<td>On specific API requests that mutate the state of resources</td>
<td>Audit Event Table in the Cloud Controller DB</td>
<td>n/a</td>
<td>PCF operator</td>
<td>31 days</td>
</tr>
<tr>
<td>• IP Address • Email address • User ID • Username</td>
<td>On each request</td>
<td>• Local VM: CEF logs • Log aggregator used by PCF operator</td>
<td>n/a</td>
<td>PCF operator</td>
<td></td>
</tr>
<tr>
<td>Audit what changes a user makes</td>
<td>On specific API requests that mutate the state of resources</td>
<td>Audit Event Table in the Cloud Controller DB</td>
<td>n/a</td>
<td>PCF operator</td>
<td></td>
</tr>
<tr>
<td>• IP Address • Email address • User ID • Username</td>
<td>On each request</td>
<td>• Local VM: CEF logs • Log aggregator used by PCF operator</td>
<td>n/a</td>
<td>PCF operator</td>
<td></td>
</tr>
<tr>
<td>Audit what user created a resource</td>
<td>When API resources are created</td>
<td>As part of the resource row in Cloud Controller DB</td>
<td>n/a</td>
<td>PCF operator</td>
<td>As long as the resource exists</td>
</tr>
</tbody>
</table>

### Routing

By default, the Gorouter logs include the X-Forwarded-For header, which may include the originating client IP. Under GDPR, client IP addresses should be considered personal data.

### Disable Client IP Logging

In Pivotal Application Service (PAS) v2.0 and later and Elastic Runtime v1.12, operators can disable logging of client IP addresses in the Gorouter.

To disable logging of client IP addresses, do the following:

1. Navigate to the Ops Manager Installation Dashboard and click the PAS or Elastic Runtime tile.
2. Click Networking.
3. Under Logging of Client IPs in CF Router, select one of the two options:
   - If the source IP address exposed by your load balancer is its own IP address, select Disable logging of X-Forwarded-For header only.
   - If the source IP address exposed by your load balancer belongs to the downstream client, select Disable logging of both source IP and X-Forwarded-For header.
4. Click Save.
5. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes** to redeploy.

### Diego

Diego is the container management system for PCF. For more information, see [Diego Components and Architecture](#).

<table>
<thead>
<tr>
<th>GDPR</th>
<th>Workflow</th>
<th>What personal data is collected?</th>
<th>When is it collected?</th>
<th>Where is it stored?</th>
<th>How is it processed?</th>
<th>Who has access to it?</th>
<th>How can I delete it?</th>
</tr>
</thead>
</table>
|      | Business Execution | Executing apps and tasks | No personal data is collected explicitly, but personal data may be encoded in app file contents or runtime metadata such as environment variables or start commands. | Runtime metadata is collected when Cloud Controller submits work specification to the Diego BBS API. File contents are collected when Diego schedules an app or a task on a Diego cell. | Runtime metadata is used to start processes inside app instance containers and to configure their environment. App file contents are presented as part of the app instance container filesystem. | Platform operators and other developers with access to the Cloud Controller space containing that app can view the data. | ✷ To delete the runtime metadata stored in the Diego BBS DB, stop the app or cancel the task that includes that data. 
✦ To delete the app file contents stored in the running app and task containers, stop the app or cancel the task to destroy the containers. To destroy the app file contents stored in the download cache on the Diego cells, recreate the Diego cell VMs. |
|      | Reports/Logs | SSH proxy logs Cloud Foundry user access. | UAA user name and ID | When the user authenticates for SSH access to an app. | The data is stored in a log file collocated with the SSH proxy instance handling the authentication request. This log file may also have its contents forwarded to a remote syslog destination. | No processing of the local log file is done automatically. If the log file contents are forwarded to a log aggregation service, they may be parsed and processed arbitrarily. | Only platform operators have access to the local log file. Platform operators or auditors may have access to these log contents in a downstream log aggregation service. | To delete the log lines containing the user ID, perform the following steps: 
1. Run `bosh recreate` on the VMs hosting the SSH proxy processes to remove all the logs on ephemeral disk. 
2. `bosh ssh` into the VMs hosting the SSH proxy processes and remove specific log lines containing user IDs. 
3. Scrub corresponding log lines from any log aggregation service. |

### Notifications Service

The Notifications Service enables operators to configure components of Cloud Foundry to send emails to end users. For more information, see [Getting Started with the Notifications Service](#).
<table>
<thead>
<tr>
<th>GDPR</th>
<th>Workflow</th>
<th>What personal data is collected?</th>
<th>When is it collected?</th>
<th>Where is it stored?</th>
<th>How is it processed?</th>
<th>Who has access to it?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Execution</strong></td>
<td>Sending email to UAA users</td>
<td>User ID</td>
<td>First email sent</td>
<td>The <code>receipts</code> table in the Notifications database</td>
<td>Stored in the Notifications database</td>
<td>Notifications operator making a database query</td>
</tr>
<tr>
<td></td>
<td>UAA user unsubscribes globally</td>
<td>User ID</td>
<td>When the UAA user unsubscribes</td>
<td>The <code>global_unsubscribes</code> table in the Notifications database</td>
<td>Stored in the Notifications database</td>
<td>Notifications operator making a database query</td>
</tr>
<tr>
<td></td>
<td>UAA user unsubscribes from a specific kind of email</td>
<td>User ID</td>
<td>When the UAA user unsubscribes</td>
<td>The <code>unsubscribes</code> table in the Notifications database</td>
<td>Stored in the Notifications database</td>
<td>Notifications operator making a database query</td>
</tr>
<tr>
<td></td>
<td>UAA user unsubscribes from a campaign in the v2 API</td>
<td>User ID</td>
<td>When the UAA user unsubscribes</td>
<td>The <code>unsubscribes</code> table in the Notifications database</td>
<td>Stored in the Notifications database</td>
<td>Notifications operator making a database query</td>
</tr>
<tr>
<td><strong>Reports/Logs</strong></td>
<td>UAA user unsubscribes</td>
<td>User email address</td>
<td>When the UAA user unsubscribes</td>
<td>Log output</td>
<td>Loggregator</td>
<td>Loggregator Firehose users</td>
</tr>
</tbody>
</table>
Security-Related PCF Tiles and Add-ons

This section provides links to other security-related services available for Pivotal Cloud Foundry (PCF) deployments.

For information about other security-related tiles provided by Pivotal partners, refer to Pivotal Network and Pivotal Documentation.

Tiles

This section provides links to security-related tiles available for Pivotal Cloud Foundry (PCF) deployments.

- **Single Sign-On (SSO)**: Provides support for native authentication, federated single sign-on, and authorization.
- **CredHub Service Broker**: Allows apps running on Pivotal Application Service (PAS) to access secure credentials in CredHub.
- **Compliance Scanner**: Enables operators and auditors to conduct compliance scanning of BOSH-managed Xenial VMs against configuration hardening guidelines.

Add-ons

This section provides links to security-related BOSH add-ons available for Pivotal Cloud Foundry (PCF) deployments.

- **File Integrity Monitoring Add-on for PCF**: Provides File Integrity Monitoring (FIM) protection within the PCF environment. This monitoring is sometimes required for compliance purposes.
- **IPsec Add-on for PCF**: Secures data transmissions inside Pivotal Cloud Foundry (PCF).
- **ClamAV Add-on for PCF**: Provides antivirus protection within the PCF environment. This protection is sometimes required for compliance purposes.
Security Guidelines for Your IaaS Provider

Pivotal Cloud Foundry supports a variety of Infrastructure as a Service (IaaS) providers. Different IaaS providers require different configuration steps to secure user data, identity information, and credentials.

Security requirements can vary broadly based on the unique configuration and infrastructure of each organization. Rather than provide specific guidance that may not apply to all use cases, Pivotal has collected links to IaaS providers’ security and identity management documentation. The documents below may help you understand how your IaaS’ security requirements impact your PCF deployment.

Pivotal does not endorse these documents for accuracy or guarantee that their contents apply to all PCF installations.

How to Use This Topic

Find your IaaS provider in the list below. The documentation items linked for each IaaS may help you configure and secure your installation infrastructure.

Amazon Web Services (AWS)

- [AWS Identity and Access Management guide](#)
  This guide is a reference for AWS’ Identity and Access Management (IAM) features. If you’re new to AWS, start here.

- [AWS identity documentation](#)
- [AWS credential documentation](#)
  This documentation provides a general definition of IAM terms and provide best practices to help you manage IaaS users and permissions.

Google Cloud Platform (GCP)

- [GCP authentication documentation](#)
  This developer-facing documentation explains general authentication guidelines for GCP.

Microsoft Azure

- [Azure security documentation](#)
  This site has documentation on Azure security tools. It provides a general guide to how to manage IaaS users and credentials.

OpenStack

- [OpenStack credential configuration](#)
- [OpenStack credential creation](#)
- [OpenStack deployment configuration](#)
  These documents provide a general reference for OpenStack service credential management.

VMware vSphere

- [vSphere Security guide (PDF)](#)
  This guide contains best practices for securing and managing a vSphere installation.
Buildpacks

Page last updated:

Buildpacks provide framework and runtime support for apps. Buildpacks typically examine your apps to determine what dependencies to download and how to configure the apps to communicate with bound services.

When you push an app, Cloud Foundry automatically detects an appropriate buildpack for it. This buildpack is used to compile or prepare your app for launch.

Note: Cloud Foundry deployments often have limited access to dependencies. This limitation occurs when the deployment is behind a firewall, or when administrators want to use local mirrors and proxies. In these circumstances, Cloud Foundry provides a Buildpack Packager app.

About Buildpacks

For general information about buildpacks, see About Buildpacks.

System Buildpacks

Cloud Foundry includes a set of system buildpacks for common languages and frameworks. This table lists the system buildpacks.

<table>
<thead>
<tr>
<th>Name</th>
<th>Supported Languages, Frameworks, and Technologies</th>
<th>GitHub Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>n/a</td>
<td>Binary source</td>
</tr>
<tr>
<td>Go</td>
<td>Go</td>
<td>Go source</td>
</tr>
<tr>
<td>HWC</td>
<td>HWC</td>
<td>HWC source</td>
</tr>
<tr>
<td>Java</td>
<td>Grails, Play, Spring, or any other JVM-based language or framework</td>
<td>Java source</td>
</tr>
<tr>
<td>.NET Core</td>
<td>.NET Core</td>
<td>.NET Core source</td>
</tr>
<tr>
<td>NGINX</td>
<td>NGINX</td>
<td>NGINX source</td>
</tr>
<tr>
<td>Node.js</td>
<td>Node or JavaScript</td>
<td>Node.js source</td>
</tr>
<tr>
<td>PHP</td>
<td>Cake, Symfony, Zend, NGINX, or HTTPD</td>
<td>PHP source</td>
</tr>
<tr>
<td>Python</td>
<td>Django or Flask</td>
<td>Python source</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R source</td>
</tr>
<tr>
<td>Ruby</td>
<td>Ruby, JRuby, Rack, Rails, or Sinatra</td>
<td>Ruby source</td>
</tr>
<tr>
<td>Staticfile</td>
<td>HTML, CSS, JavaScript, or NGINX</td>
<td>Staticfile source</td>
</tr>
</tbody>
</table>
Community Buildpacks

You can find a list of unsupported, community-created buildpacks here: cf-docs-contrib.

Customizing and Developing Buildpacks

For information about customizing existing buildpacks and developing new buildpacks, see Customizing and Developing Buildpacks.

For information about updating and releasing a new version of a Cloud Foundry buildpack through the Cloud Foundry Buildpacks Team Concourse pipeline, see Using CI for Buildpacks. You can use this as a model when working with Concourse to build and release new versions of your own buildpacks.
About Buildpacks

Page last updated:

This topic provides links to additional information about using buildpacks. Each of the following are applicable to all supported buildpack languages and frameworks:

- Buildpacks
- Stack Association
- Pushing an Application with Multiple Buildpacks
- Using a Proxy
- Supported Binary Dependencies
- Production Server Configuration
Buildpacks

Page last updated:

This topic describes how buildpacks work in Cloud Foundry.

Buildpack Scripts

A buildpack repository may contain the following five scripts in the `bin` directory:

- **bin/detect**: determines whether or not to apply the buildpack to an app.
- **bin/supply**: provides dependencies for an app.
- **bin/finalize**: prepares the app for launch.
- **bin/release**: provides feedback metadata to Cloud Foundry indicating how the app should be executed.
- **bin/compile**: is a deprecated alternative to `bin/supply` and `bin/finalize`.

The `bin/supply` and `bin/finalize` scripts replace the deprecated `bin/compile` script. Older buildpacks may still use `bin/compile` with the latest version of Cloud Foundry. In this case, applying multiple buildpacks to a single app is not supported. Similarly, newer buildpacks may still provide `bin/compile` for compatibility with Heroku and older versions of Cloud Foundry.

The `bin/gemfile` script is required for non-final buildpacks. The `bin/finalize` (or `bin/compile`) script is required for final buildpacks.

---

### Note:
In this document, the terms **non-final buildpack** and **final buildpack**, or **last buildpack**, are used to describe the process of applying multiple buildpacks to an app. See the following example:

```
cf push APP-NAME -b FIRST-BUILDPACK -b SECOND-BUILDPACK -b FINAL-BUILDPACK
```

---

### Note:
If you use only one buildpack for your app, this buildpack behaves as a final, or last, buildpack.

---

### Note:
When using multi-buildpack support, the last buildpack in order is the final buildpack, and is able to make changes to the app and determine a start command. All other specified buildpacks are non-final and only supply dependencies.

---

**bin/detect**

The `detect` script determines whether or not to apply the buildpack to an app. The script is called with one argument, the `build` directory for the app. The `build` directory contains the app files uploaded when a user performs a `cf push`.

The `detect` script returns an exit code of `0` if the buildpack is compatible with the app. In the case of system buildpacks, the script also prints the buildpack name, version, and other information to `STDOUT`.

The following is an example `detect` script that checks for a Ruby app based on the existence of a `Gemfile`:

```bash
#!/usr/bin/env ruby

gemfile_path = File.join ARGV[0], "Gemfile"

if File.exist?(gemfile_path)
  puts "Ruby"
  exit 0
else
  exit 1
end
```

Optionally, the buildpack `detect` script can output additional details provided by the buildpack developer. This includes buildpack versioning information and a list of configured frameworks and their associated versions.

The following is an example of the detailed information returned by the Java buildpack:

```
java-buildpack=v3.0-https://github.com/cloudfoundry/java-buildpack.git#3bd15e1
openjdk-jre=1.8.0_45
spring-auto-configuration=1.7.0_RELEASE
tomcat-access-logging-support=2.4.0_RELEASE
tomcat-in...
```
For more information, see the Buildpack Detection section below.

bin/supply

The `supply` script provides dependencies for the app and runs for all buildpacks. All output sent to `STDOUT` is relayed to the user through the Cloud Foundry Command Line Interface (cf CLI).

The script is run with four arguments:

- The `build` directory for the app
- The `cache` directory, which is a location the buildpack can use to store assets during the build process
- The `deps` directory, which is where dependencies provided by all buildpacks are installed
- The `index`, which is a number that represents the ordinal position of the buildpack

The `supply` script stores dependencies in `deps/index`. It may also look in other directories within `deps` to find dependencies supplied by other buildpacks.

The `supply` script must not modify anything outside of the `deps/index` directory. Staging may fail if such modification is detected.

The `cache` directory provided to the `supply` script of the final buildpack is preserved even when the buildpack is upgraded or otherwise changes. The `finalize` script also has access to this cache directory.

The `cache` directories provided to the `supply` scripts of non-final buildpacks are cleared if those buildpacks are upgraded or otherwise change.

The following is an example of a simple `supply` script:

```ruby
#!/usr/bin/env ruby

STDOUT.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]
deps_path = ARGV[2]
index = ARGV[3]

install_ruby

private
def install_ruby
  puts "Installing Ruby"

  # !!! build tasks go here !!!
  # download ruby
  # install ruby
end
```

bin/finalize

The `finalize` script prepares the app for launch and runs only for the last buildpack. All output sent to `STDOUT` is relayed to the user through the cf CLI.

The script is run with four arguments:

- The `build` directory for the app
- The `cache` directory, which is a location the buildpack can use to store assets during the build process
- The `deps` directory, which is where dependencies provided by all buildpacks are installed
- The `index`, which is a number that represents the ordinal position of the buildpack

The `finalize` script may find dependencies installed by the `supply` script of the same buildpack in `deps/index`. It may also look in other directories within `deps` to find dependencies supplied by other buildpacks.

Note: Cloud Foundry detects only one buildpack by default. When multiple buildpacks are desired, you must explicitly specify them.
The cache directory provided to the finalize script is preserved even when the buildpack is upgraded or otherwise changes. The supply script of the same buildpack also has access to this cache directory.

The following is an example of a simple finalize script:

```ruby
#!/usr/bin/env ruby

#sync output
$stdout.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]
deps_path = ARGV[2]
index = ARGV[3]

setup_ruby

private
def setup_ruby
  puts "Configuring your app to use Ruby"

  # ! ! ! build tasks go here ! ! !
  # setup ruby

end
```

bin/compile (Deprecated)

The compile script is deprecated. It encompasses the behavior of the supply and finalize scripts for single buildpack apps by using the build directory to store dependencies.

The script is run with two arguments:

- The build directory for the app
- The cache directory, which is a location the buildpack can use to store assets during the build process

During the execution of the compile script, all output sent to STDOUT is relayed to the user through the cf CLI.

bin/release

The release script provides feedback metadata to Cloud Foundry indicating how the app should be executed. The script is run with one argument, the build directory. The script must generate a YAML file in the following format:

```yaml
default_process_types:
  web: start_command.filetype
```

```
default_process_types
  web: start_command.filetype
```

indicates the type of app being run and the command used to start it. This start command is used if a start command is not specified in the cf push or in a Procfile.

At this time, only the web type of apps is supported.

Note: To define environment variables for your buildpack, add a Bash script to the .profile.d directory in the root folder of your app.

The following example shows what a release script for a Rack app might return:

```
default_process_types:
  web: bundle exec rackup config.ru -p $PORT
```

Note: The web command runs as bash -c COMMAND when Cloud Foundry starts your app. Refer to the command attribute section for more information about custom start commands.
Droplet Filesystem

The buildpack staging process extracts the droplet into the `/home/vcap` directory inside the instance container and creates the following filesystem tree:

```
app/
deps/
logs/
tmp/
staging_info.yml
```

The `app` directory includes the contents of the `build` directory, and `staging_info.yml` contains the staging metadata saved in the droplet.

Buildpack Detection

When you push an app, Cloud Foundry uses a detection process to determine a single buildpack to use. For general information about this process, see [How Apps Are Staged](#).

During staging, each buildpack has a position in a priority list. You can retrieve this position by running `cf buildpacks`.

Cloud Foundry checks if the buildpack in position 1 is a compatible buildpack. If the position 1 buildpack is not compatible, Cloud Foundry moves on to the buildpack in position 2. Cloud Foundry continues this process until the correct buildpack is found.

If no buildpack is compatible, the `cf push` command fails with the following error:

```
None of the buildpacks detected a compatible application
Exit status 222
Staging failed: Exited with status 222
FAIL
NoAppDetectedError
```

For a more detailed account of how Cloud Foundry interacts with the buildpack, see the [Sequence of Interactions](#) section below.

Sequence of Interactions

This section describes the sequence of interactions between the Cloud Foundry platform and the buildpack. The sequence of interactions differs depending on whether the platform skips or performs buildpack detection.

No Buildpack Detection

Cloud Foundry skips buildpack detection if the developer specifies one or more buildpacks in the app manifest or in the `cf push APP-NAME -b BUILDPACK-NAME` cf CLI command.

If you explicitly specify buildpacks, Cloud Foundry performs the following interactions:

1. For each buildpack except the last buildpack, the platform does the following:
   a. Creates the `deps/index` directory
   b. Runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`
   c. Accepts any modification of the `deps/index` directory
   d. Accepts any modification of the `cache` directory
   e. May disallow modification of any other directories

2. For the last buildpack, the platform does the following:
   a. If `/bin/finalize` is present:
      i. Creates the `deps/index` directory if it does not exist
      ii. If `/bin/supply` is present, runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`
      iii. Accepts any modification of the `deps/index` directory

© Copyright Pivotal Software Inc, 2013-2019
iv. May disallow modification of the `build` directory
v. Runs `/bin/finalize` with the `build`, `cache`, and `deps` directories and the buildpack `index` directory
vi. Accepts any modification of the `build` directory

b. If `/bin/finalize` is not present:
   i. Runs `/bin/compile` with the `build` and `cache` directories
   ii. Accepts any modification of the `build` directory

b. If `/bin/finalize` is not present:
   i. Runs `/bin/compile` with the `build` and `cache` directories
   ii. Accepts any modification of the `build` directory
   
At the end of this process, the `deps` directory is included at the root of the droplet, adjacent to the `app` directory.

Buildpack Detection

Cloud Foundry performs buildpack detection if the developer does not specify one or more buildpacks in the app manifest or in the `cf push APP-NAME-b BUILDPACK-NAME` CLI command.

Note: Cloud Foundry detects only one buildpack to use with the app.

If the platform performs detection, it does the following:

1. Runs `/bin/detect` for each buildpack
2. Selects the first buildpack with a `/bin/detect` script that returns a zero exit status
3. If `/bin/finalize` is present:
   a. Creates the `deps` `/index` directory if it does not exist
   b. If `/bin/supply` is present, runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index` directory
   c. Accepts any modification of the `deps` `/index` directory
   d. May disallow modification of the `build` directory
   e. Runs `/bin/finalize` on the `build`, `cache`, and `deps` directories
   f. Accepts any modification of the `build` directory
4. If `/bin/finalize` is not present:
   a. Runs `/bin/compile` on the `build` and `cache` directories
   b. Accepts any modification of the `build` directory
5. Runs `/bin/release` to determine staging information

At the end of this process, the `deps` directory is included at the root of the droplet, adjacent to the `app` directory.
Stack Association

This topic describes the stack association feature for Pivotal Cloud Foundry (PCF) buildpacks.

Note: This functionality is supported as of CAPI v1.58.0 and cf CLI v6.39.0.

Warning: To avoid security exposure, ensure that you migrate your apps and custom buildpacks to use the cflinuxfs3 stack based on Ubuntu 18.04 (Bionic Beaver). cflinuxfs2 is based on Ubuntu 14.04 (Trusty Tahr), which reaches end of general support (EOGS) in April 2019.

Overview

Each buildpack in your PCF deployment is associated with a stack. You can see this when you run `cf buildpacks`: there is a stack column in the output that shows a corresponding stack for each buildpack. See the following example:

```
$ cf buildpacks
Getting buildpacks...

<table>
<thead>
<tr>
<th>buildpack</th>
<th>position</th>
<th>enabled</th>
<th>locked</th>
<th>filename</th>
<th>stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>staticfile_buildpack</td>
<td>1</td>
<td>true</td>
<td>false</td>
<td>staticfile_buildpack Cached-cflinuxfs3-v1.4.29.zip</td>
<td>cflinuxfs3</td>
</tr>
<tr>
<td>java_buildpack_offline</td>
<td>2</td>
<td>true</td>
<td>false</td>
<td>java-buildpack-offline-cflinuxfs3-v4.12.1.zip</td>
<td>cflinuxfs3</td>
</tr>
<tr>
<td>ruby_buildpack</td>
<td>3</td>
<td>true</td>
<td>false</td>
<td>ruby_buildpack Cached-cflinuxfs3-v1.7.21.zip</td>
<td>cflinuxfs3</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Because of this stack association, buildpacks do not have to be uniquely named. This helps in managing similar buildpacks that are compatible with different stacks.

The `buildpack packager` includes a `-stack` option. If you use this option and upload a buildpack to PCF, the Cloud Controller detects the stack association and creates a stack record for the buildpack.

Buildpacks without a Stack Record

Some buildpacks may have a missing stack record. For example, if you uploaded a custom buildpack before PCF introduced stack association. The output of `cf buildpacks` shows a blank stack column if the buildpack does not have a stack record.

In this case, you must manually assign a stack to the buildpack. To do this, run:

```
$ cf update-buildpack BUILDPACK-NANE --assign-stack stack
```

Buildpacks with a missing stack record will continue to work, but are more manageable when the stack record is present.

Consider the following behavior when pushing apps to a deployment that has buildpacks with a missing stack record:

- If you push an app and specify a stack with `cf push app-name -s stack`, PCF uses that stack. Otherwise, it uses the system default, cflinuxfs3.
- You may see additional logging in the buildpack detection output of the `cf push` command when PCF detects buildpacks without a stack record.

Managing Stack Association with the cf CLI

The cf CLI commands for managing buildpacks include functionality to support association between buildpacks and stacks. The `update-buildpack`, `rename-buildpack`, and `delete-buildpack` commands all include a `-s` flag for specifying a stack.

When operating on buildpacks with the cf CLI, consider the following:

- You cannot upload a buildpack with `cf create-buildpack` if a buildpack of the same name already exists and it has a missing stack record.
- When using `cf create-buildpack`, you may inadvertently create a duplicate buildpack with a nil stack. `cf create-buildpack` does not disallow creation of buildpacks with no stack association.
- The `-s` flag is required when there are two buildpacks with the same name. If you are operating on a uniquely named buildpack, you do not need to specify its stack.
If you have two buildpacks of the same name, one with a stack record and one without, running cf CLI commands without \(-s\) operates on the buildpack with the missing stack record.

Example Scenarios

See the following examples of managing buildpacks with the cf CLI, which are applicable when running `cf update-buildpack`, `cf rename-buildpack`, or `cf delete-buildpack`:

- **Updating, renaming, or deleting a uniquely-named buildpack:**
  - You have a single buildpack named `my-buildpack`, and it is associated with `stack_a`. If you want to delete the buildpack, you can run:
    ```
    cf delete-buildpack my-buildpack
    ```
    You can also provide \(-s stack_a\), but the option is not required if you have a uniquely-named buildpack.

- **Updating, renaming, or deleting a uniquely-named buildpack that has a `nil` stack:**
  - You have a single buildpack named `my-buildpack`, and it is not associated with a stack. If you want to delete the buildpack, you can run:
    ```
    cf delete-buildpack my-buildpack
    ```

- **Updating, renaming, or deleting a buildpack when another buildpack exists with the same name, and both buildpacks have stack associations:**
  - You have two buildpacks named `my-buildpack`, one that is associated with `stack_a` and the other associated with `stack_b`. If you want to delete the buildpack that uses `stack_a`, you can run:
    ```
    cf delete-buildpack my-buildpack -s stack_a
    ```

- **Updating, renaming, or deleting a buildpack when another buildpack exists with the same name. One buildpack has a stack association, and the other buildpack has a `nil` stack:**
  - You have two buildpacks named `my-buildpack`, one associated with `stack_a` and the other associated with no `nil` stack association:
    - If you want to delete the buildpack that uses `stack_a`, you can run:
      ```
      cf delete-buildpack my-buildpack -s stack_a
      ```
    - If you want to delete the buildpack that is associated with the `nil` stack, run:
      ```
      cf delete-buildpack my-buildpack
      ```
Pushing an Application with Multiple Buildpacks

Page last updated:

This topic describes how developers can push an application with multiple buildpacks.

Note: As an alternative to the cf CLI procedure below, you can specify multiple buildpacks in your application manifest. This is not compatible with Deprecated App Manifest Features.

For more information about pushing applications to Pivotal Application Service (PAS), see the Deploying an Application topic.

Specifying Buildpacks with the cf CLI

To push an application with multiple buildpacks using the Cloud Foundry Command Line Interface (cf CLI), perform the following procedure:

Note: You must use cf CLI v6.38 or later.

1. Run the following command to ensure that you are using the cf CLI v6.38 or later:

   ```bash
   $ cf version
   ```

   For more information about upgrading the cf CLI, see Installing the cf CLI.

2. To push your app with multiple buildpacks, specify each buildpack with a `--buildpack` flag:

   ```bash
   $ cf push YOUR-APP --buildpack BUILDPACK-NAME-1 --buildpack BUILDPACK-NAME-2 ... --buildpack FINAL-BUILDPACK-NAME
   ```

   The last buildpack you specify is the final buildpack, which can modify the launch environment and set the start command.

   To see a list of available buildpacks, run `cf buildpacks`.

For more information on multi-buildpack order, see the Buildpacks topic.

For more information about using the cf CLI, see the Cloud Foundry Command Line Interface topic.
Using a Proxy

This topic describes how developers can use a proxy with the buildpacks for their application.

Use a Proxy

Buildpacks can use proxies using the `http_proxy` and `https_proxy` environment variables. You should set these to the proxy hostname or port.

All of the buildpacks automatically use these proxy environment variables correctly. If any buildpacks contacts the Internet during staging, it does so through the proxy host. The binary buildpack does not use a proxy because it does not use the Internet during staging.

To set a proxy for buildpacks to use during staging, perform one of the following procedures:

- Set the environment variables by adding the following section to the `env` block of the application manifest:

```yaml
---
env:
  http_proxy: http://YOUR-HTTP-PROXY:PORT
  https_proxy: https://YOUR-HTTPS-PROXY:PORT
```

- Set the environment variables with the Cloud Foundry Command Line Interface (cf CLI) using the `cf set-env` command:

```bash
```

**Note:** While many apps use the `http_proxy` and `https_proxy` environment variables at runtime, some do not. The buildpack does not add extra functionality to make proxies work at runtime.

© Copyright Pivotal Software Inc, 2013-2019
Supported Binary Dependencies

Page last updated:

Each buildpack only supports the stable patches for each dependency listed in the buildpack's `manifest.yml` and also in its GitHub releases page. For example, see the [php-buildpack releases page](#).

If you try to use an unsupported binary, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:
...?
! exit?
Staging failed: Buildpack compilation step failed
```
Production Server Configuration

This topic describes how to configure a production server for your apps.

When you deploy an app, PAS determines the command used to start the app through the following process:

1. If the developer uses the command `cf push -c COMMAND`, then PAS uses `COMMAND` to start the app.

2. If the developer creates a file called a Procfile, PAS uses the Procfile to configure the command that launches the app. See the About Procfiles section below for more information.

3. If the developer does not use `cf push -c COMMAND` and does not create a Procfile, then PAS does one of the following, depending on the buildpack:
   - Uses a default start command.
   - Fails to start the app and shows a warning that the app is missing a Procfile.

About Procfiles

One reason to use a Procfile is to specify a start command for buildpacks where a default start command is not provided. Some buildpacks, such as Python, that work on a variety of frameworks, do not attempt to provide a default start command.

Another reason to use a Procfile is to configure a production server for web apps.

A Procfile enables you to declare required runtime processes, called process types, for your web app. Process managers in a server use the process types to run and manage the workload. In a Procfile, you declare one process type per line and use the following syntax:

```
PROCESS_TYPE: COMMAND
```

- `PROCESS_TYPE` is `web`. A `web` process handles HTTP traffic.
- `COMMAND` is the command line to launch the process.

For example, a Procfile with the following content starts the launch script created by the build process for a Java app:

```
web: build/install/MY-PROJECT-NAME/bin/MY-PROJECT-NAME
```

Specify a Web Server

Follow these steps to specify a web server using a Procfile. For more information about configuring a web server for Rails apps, see the Configure a Ruby Web Server section of this topic.

1. Create a blank file with a command line for a `web` process type.
2. Save it as a file named `Procfile` with no extension in the root directory of your app.
3. Push your app.

Configure a Ruby Web Server

PAS uses the default standard Ruby web server library WEBrick for Ruby and Ruby on Rails apps. However, PAS can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn.

To instruct PAS to use a web server other than WEBrick, perform the following steps:

1. Add the gem for the web server to your Gemfile.
2. In the `config` directory of your app, create a new configuration file or modify an existing file. Refer to your web server documentation for how to configure this file. The following example uses the Puma web server:
In the root directory of your app, create a Procfile and add a command line for a `web` process type that points to your web server. For information about configuring the specific command for a process type, see your web server documentation.

The following example shows a command that starts a Puma web server and specifies the app runtime environment, TCP port, and paths to the server state information and configuration files:

```
web: bundle exec puma -e $RAILS_ENV -p 1234 -S ~/puma -C config/puma.rb
```
Binary Buildpack

Use the binary buildpack for running arbitrary binary web servers.

Push an App

Specify the binary buildpack to stage an app as a binary file. On a command line, use `cf push APP-NAME` with the `-b` option to specify the buildpack.

For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/binary-buildpack.git
```

You can provide Cloud Foundry with the shell command to execute your binary in the following two ways:

- **Procfile:** In the root directory of your app, add a `Procfile` that specifies a `web` task:

```
web: ./app
```

- **Command line:** Use `cf push APP-NAME` with the `-c` option:

```
$ cf push my_app -c './app' -b binary_buildpack
```

Compile your Binary

Cloud Foundry expects your binary to bind to the port specified by the `PORT` environment variable.

The following example in Go binds a binary to the PORT environment variable:

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprint(w, "Hello, %s", "world!")
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":4000", nil)
}
```

Your binary should run without any additional runtime dependencies on the cflinuxfs3 or lucid64 root filesystem (rootfs). Any such dependencies should be statically linked to the binary.

To boot a Docker container running the cflinuxfs3 filesystem, run the following command:

```
$ docker run -it cloudfoundry/cflinuxfs3 bash
```

To boot a Docker container running the lucid64 filesystem, run the following command:

```
$ docker run -it cloudfoundry/lucid64 bash
```

To compile the above Go application on the rootfs, golang must be installed. `apt-get install golang` and `go build app.go` will produce an `app` binary.

When deploying your binary to Cloud Foundry, use `cf push` with the `-s` option to specify the root filesystem it should run against.
BOSH Configured Custom Trusted Certificate Support

Your platform operator can configure the platform to add the custom certificates into the application container. The custom trusted certificates are added to the `/etc/ssl/certs` directory and can be used by binary applications.

For more information, see Configuring Trusted System Certificates for Applications.

.NET Apps

You can use the Binary buildpack to push .NET Core apps, .NET Console apps, and self-hosted apps to the Windows stack. Self-hosted apps are apps that include their own web server.

To push a .NET Console app using the binary buildpack, see Console Applications in the .NET Cookbook for more information.

To deploy .NET Core apps to Windows using the binary buildpack, build it as a self-hosted app and push it with the binary buildpack. Note that you will need to specify the `windows` stack and use a custom start command in your manifest file. Here is an example manifest:

```
---
applications:
  - name: mydemoapp
    random-route: true
    stack: windows
    health-check-type: none
    command: cmd /c .CloudFoundry --server.urls http://*:%PORT%
    env:
      ASPNETCORE_ENVIRONMENT: production
```

For information about deploying different types of .NET apps, follow the links in the table below.

<table>
<thead>
<tr>
<th>Type of .NET App</th>
<th>Buildpack</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP.NET MVC</td>
<td>HWC</td>
</tr>
<tr>
<td>ASP.NET Web Forms</td>
<td></td>
</tr>
<tr>
<td>ASP.NET WebAPI Apps</td>
<td></td>
</tr>
<tr>
<td>Windows Communication Foundation (WCF)</td>
<td></td>
</tr>
<tr>
<td>.NET Core pushed to Linux stack</td>
<td>.NET Core</td>
</tr>
</tbody>
</table>

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the binary buildpack in Cloud Foundry, see the binary-buildpack GitHub repository.

You can find current information about this buildpack on the binary buildpack release page in GitHub.
Go Buildpack

Page last updated:

Supported Versions

Supported Go versions can be found in the release notes.

Push an App

The Go buildpack will be automatically detected in the following circumstances:

- Your app has been packaged with `godep` using `godep save`.
- Your app has a `vendor/` directory and has any files ending with `.go`.
- Your app has a `GOPACKAGENAME` environment variable specified and has any files ending with `.go`.
- Your app has a `glide.yml` file and is using `glide`, starting in buildpack version 1.7.9.
- Your app has a `Gopkg.toml` file and is using `dep`, starting in buildpack version 1.8.9.

If your Cloud Foundry deployment does not have the Go Buildpack installed, or the installed version is out of date, you can use the latest version with the command:

```
$ cf push my_app -h https://github.com/cloudfoundry/go-buildpack.git
```

When specifying versions, specify only major/minor versions, such as Go 1.6, rather than Go 1.6.0. This ensures you receive the most recent patches.

Start Command

When pushing Go apps, you can specify a start command for the app. You can place the start command in the `Procfile` file in root directory of your app.

For example, if the binary generated by your Go project is `my-go-server`, your `Procfile` could contain the following:

```
web: my-go-server
```

For more information about Procfiles, see the Configuring a Production Server topic.

You can also specify the start command for your app in the `manifest.yml` file in the root directory. For example, your `manifest.yml` could contain the following:

```
---
applications:
  - name: my-app-name
    command: my-go-server
```

If you do not specify a start command in a `Procfile`, in the manifest, or with the `-c` flag for `cf push`, the generated binary will be used as the start command. Example: `my-go-server`

Push an App with godep

If you are using `godep` to package your dependencies, make sure that you have created a valid `Godeps/Godeps.json` file in the root directory of your app by running `godep save`.

When using godep, you can fix your Go version in `GoVersion` key of the `Godeps/Godeps.json` file.

Go Sample App
Sample Go app

An example Godeps/Godeps.json:

```json
{
  "ImportPath": "go_app",
  "GoVersion": "go1.6",
  "Deps": []
}
```

### Push an App with Glide

If you use glide to specify or package your dependencies, make sure that you have created a valid glide.yml file in the root directory of your app by running:

```
$ glide init
```

To vend your dependencies before pushing, run:

```
$ glide install
```

This will generate a vendor directory and a glide.lock file specifying the latest compatible versions of your dependencies. You must have a glide.lock file when pushing a vendored app. You do not need a glide.lock file when deploying a non-vendored app.

#### Glide

Sample Go app with Glide

An example glide.yml file:

```yaml
package: go_app_with_glide
import:
  - package: github.com/ZiCog/shiny-thing
    subpackages:
      - foo
```

You can specify Go version in the manifest.yml file:

```yaml
---
applications:
  - name: my-app-name
    env:
      GOVERSION: go1.8
```

### Push an App with dep

If you use dep to specify or package your dependencies, make sure that you have created a valid Gopkg.toml file in the root directory of your app by running:

```
$ dep init
```

To vend your dependencies before pushing, run:

```
$ dep ensure
```

This will generate a vendor directory and a Gopkg.lock file specifying the latest compatible versions of your dependencies. You must have a Gopkg.lock file when pushing a vendored app. You do not need a Gopkg.lock file when deploying a non-vendored app.

#### dep

Sample Go app with dep

An example Gopkg.toml file:

```toml
[[constraint]]
branch = "master"
name = "github.com/ZiCog/shiny-thing"
```

You can specify Go version in the manifest.yml file:
Push an App with Native Go Vendoring

If you use the native Go vendoring system, which packages all local dependencies in the `vendor/` directory, you must specify your app’s package name in the `GOPACKAGENAME` environment variable.

An example `manifest.yml`:

```yaml
---
applications:
  - name: my-app-name
    env:
      GOVERSION: go1.8
```

Go 1.6

- [Sample Go 1.6 app with native vendoring](#)

Go 1.6 has vendoring enabled by default. Set the `GO15VENDOREXPERIMENT` environment variable to `0` to disable vendoring.

An example `manifest.yml` file:

```yaml
---
applications:
  - name: my-app-name
    command: go-online
    env:
      GOPACKAGENAME: example.com/user/app-package-name
```

Go 1.7 and Later

- [Sample Go 1.7 app with native vendoring](#)

Go 1.7 and later always has vendoring enabled, and you cannot disable it with an environment variable.

An example `manifest.yml`:

```yaml
---
applications:
  - name: my-app-name
    command: example-project
    env:
      GOVERSION: go1.6
      GOPACKAGENAME: example.com/user/app-package-name
```

Pass a Symbol and String to the Linker

The Go buildpack supports the Go [linker's `-X symbol value` ability](#), to set the value of a string at link time. Set the `GO_LINKER_SYMBOL` and `GO_LINKER_VALUE` in the application's configuration before pushing code.

This can be used to embed the commit SHA or other build-specific data directly into the compiled executable.

For a sample Go app, see the [go-buildpack](#) repository on GitHub.
C Dependencies

The Go buildpack supports building with C dependencies using cgo. You can set config vars to specify cgo flags to, for example, specify paths for vendored dependencies. As an example, to build `gopgsqldriver`, add the config var `CGO_CFLAGS` with the value `-I/app/code/vendor/include/postgresql` and include the relevant Postgres header files in `/vendor/include/postgresql` in your app.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see [Using a Proxy](#).

BOSH Configured Custom Trusted Certificate Support

Go uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Go buildpack in Cloud Foundry, see the [go-buildpack GitHub repository](#).

You can find current information about this buildpack on the Go buildpack [release page](#) in GitHub.
This topic describes how to configure your .NET Framework apps for use with the HWC buildpack and how to push your .NET Framework apps to Pivotal Application Service for Windows (PASW).

**Prerequisites**

Using the HWC buildpack requires deploying Windows cells with PASW.

The HWC buildpack supports the following common app types by default:

- ASP.NET MVC
- ASP.NET Web Forms
- ASP.NET WebAPI Apps
- Windows Communication Foundation (WCF)

For information about deploying different types of .NET apps, follow the links in the table below.

<table>
<thead>
<tr>
<th>Type of .NET App</th>
<th>Buildpack</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET Console</td>
<td>Binary</td>
</tr>
<tr>
<td>.NET Core pushed to Linux stack</td>
<td>.NET Core</td>
</tr>
<tr>
<td>.NET Core pushed to Windows stack</td>
<td>Binary</td>
</tr>
</tbody>
</table>

**Overview**

The HWC buildpack provides a runtime server that uses the Hosted Web Core API for running .NET Framework applications in Windows Server containers. For more information, see [Hosted Web Core API Reference](http://docs.pivotal.io) in the Microsoft documentation.

The HWC buildpack provides access to .NET Framework 4.5.1 and later, made available by the Windows root file system (rootsfs).

Before you push your first app using the HWC buildpack, see the [Getting Started](http://docs.pivotal.io) guide in the .NET Cookbook.

**Step 1. Configure HWC**

HWC relies on `Web.config` and `applicationHost.config` configuration files for configuring the .NET applications.

Most `Web.config` files work out of the box with PASW, with the following constraints:

- Integrated Windows Authentication (IWA) is not yet supported on PASW.
- SQL server connection strings must use fully qualified domain names.
- Place connection string values in environment variables or `user-provided service instances`.

Additionally, `Shadow Copy Setting` and `Dynamic and Static HTTP Compression` `Web.config` settings can be customized as needed.

The HWC buildpack includes a default configuration for the `applicationHost.config`, similar to IIS.

**Step 2. Add a Global Error Handler**

Before you push your app for the first time, add a global error handler to receive log information from your app if it crashes on startup.

To configure a global error handler that logs to `stdout`, see [Application Error Handling](http://docs.pivotal.io) in the .NET Cookbook.
Step 3. Push an App

Follow the steps below to push your application.

1. Build your HWC app in Visual Studio.

2. On the command line, navigate to the directory containing the app files.

3. To push your HWC app, run the following `cf push` command:

   ```shell
   cf push APP-NAME -s windows -b hwc_buildpack
   ```

   Where `APP-NAME` is the name you want to give your app.

   For example:

   ```shell
   $ cf push my-app -s windows -b hwc_buildpack
   Creating app my-app in org sample-org / space sample-space as username@example.com...
   OK
   ...
   requested state: started
   instances: 1/1
   usage: 1 GB x 1 instances
   urls: my-app.example.com
   ```

4. Confirm your application is running by finding your app's URL in the push command output and navigating to it. In the example above, `my-app.example.com` is the URL of your app.

Features

The following features can be used with HWC buildpack:

- **Context Path Routing**
- **Shadow Copy Setting**
- **Dynamic and Static HTTP Compression**
- **URL Rewrite**
- **Profile Scripts**

Context Path Routing

Context path routing enables multiple apps to share the same route hostname, such as `app1.example.com/app2`. The context path routing feature is analogous to IIS virtual directories.

Making an application accessible under another app's URL requires pushing both apps and applying a map-route between them. To define a context path route, such as `app1.example.com/app2`, run the following commands:

1. To push the primary app, run the following command:

   ```shell
   cf push TOP-LEVEL-APP-NAME -s windows -b hwc_buildpack
   ```

   Where:
   - `TOP-LEVEL-APP-NAME` is your top-level app's name.

2. To push the secondary app and disable the app's starting and default routing, run the following command:

   ```shell
   cf push LOWER-LEVEL-APP-NAME --no-start --no-route -s windows -b hwc_buildpack
   ```

   Where:
   - `LOWER-LEVEL-APP-NAME` is your lower-level app's name.

3. To map routes between the primary and secondary apps, run the following command:
4. To start the secondary app, run the following command:

```
cf start LOWER-LEVEL-APP-NAME
```

Where: **LOWER-LEVEL-APP-NAME** is your lower-level app’s name.

For example, the following commands define context path routing for two HWC apps, `app1` and `app2`, where `app2` is made accessible under `app1` as `app1.example.com/app2`:

```
cf push app1 -s windows -b hwc_buildpack
# if you have set CF_INSTANCE to a specific instance number
# cf push app1 -s windows -b hwc_buildpack -i $CF_INSTANCE

cf push app2 --no-start --no-route -s windows -b hwc_buildpack

cf map-route app2 example.com --hostname app1 --path app2

cf start app2
```

**Note:** HWC-hosted apps use the [VCAP_APPLICATION](#) environment variable to read out the bound app URIs. Any context path that exists underneath the root in the app’s bound route corresponds to the [applicationHost.config](#) file.

**Shadow Copy Setting**

Shadow Copy is a hosting option that copies assemblies for an app in the `bin` directory to the app’s temporary files directory. This feature is turned off and unnecessary for apps running under Cloud Foundry. An app can override this setting in its [Web.config](#) file.

**Dynamic and Static HTTP Compression**

The HWC buildpack enables dynamic and static HTTP compression by default. You can disable HTTP compression in your app’s [Web.config](#) file.

Dynamic HTTP compression is hardcoded at level 4. Static HTTP compression is hardcoded at level 9.

**URL Rewrite**

The HWC buildpack supports the URL Rewrite module. It is preinstalled in the Windows file system.

**Profile Scripts**

The HWC buildpack allows developers to provide [profile.bat](#) scripts with their applications. You can use a [profile.bat](#) script to perform app-specific initialization tasks, such as setting custom environment variables.

For information about configuring [profile.bat](#) scripts, see the Configure Pre-Runtime Hooks section of Deploying an Application.

**Buildpack Support**

A number of channels exist to assist you with using the HWC buildpack, or when developing your own HWC buildpack.

- **HWC Buildpack Repository in GitHub**: Find more information about using and extending the HWC buildpack in the HWC buildpack repository in GitHub.
- **Release Notes**: Find current information about this buildpack on the HWC buildpack release page in GitHub.
- **Slack**: Join the #buildpacks channel in the Cloud Foundry Slack community.
Creating an Extension Buildpack for .NET Apps

This topic explains how to write extension buildpacks for .NET apps.

Prerequisites

- Linux or MacOS development machine or VM
- `golang` v1.10 or later programming language
- `direnv` environment variable manager for your shell

Overview

Most .NET Framework apps are pushed with the Hosted Web Core (HWC) buildpack.

The HWC buildpack contains the commonly required .NET dependencies used by .NET apps. This set of HWC buildpack dependencies is a subset entire gamut of possible .NET dependencies and does not contain all the DLLs that every app could need.

An extension buildpack supplies a custom set of .NET dependencies to an app’s container.

You cannot add modules or extensions directly to the HWC buildpack. If the HWC buildpack lacks dependencies your app requires, we recommend that you create an extension buildpack containing the missing dependencies required by your app.

Note: The HWC buildpack contains a number of built-in HWC features, for example, the URL Rewrite and HTTP compression modules. Extensions should only provide additional functionality to your app. For more information about existing HWC features, see Features in HWC Buildpack.

For additional extension buildpack configuration and creation commands see: Creating a Custom Buildpack in the Pivotal Engineering Journal.

Step 1. Initialize an Extension Buildpack

The `buildpack-packager` provides boilerplate code that you can use to start writing your buildpack.

1. To install the `buildpack-packager` CLI tool, run the commands below.

   ```bash
   go get github.com/cloudfoundry/libbuildpack/
goinstall github.com/cloudfoundry/libbuildpack/tree/master/packager
   ```

2. To create your buildpack boilerplate, run the following command:

   ```bash
   buildpack-packager init --name BUILDPACK-NAME --path BUILDPACK-DIRECTORY
   ```

   Where:

   - `BUILDPACK-NAME` is the name of the buildpack you are creating.
   - `BUILDPACK-DIRECTORY` is the directory where the boilerplate code will be written. If the directory does not exist, it is created by the `buildpack-packager init` process.

   For example:

   ```bash
   $ buildpack-packager init --name my-example --path=my-example-buildpack
   ```

3. To activate direnv in your buildpack directory, run the following commands:

   ```bash
   cd BUILDPACK-DIRECTORY
direnv allow
   ```

   Where: `BUILDPACK-DIRECTORY` is the directory created by the buildpack-packager command, containing the boilerplate code.
Step 2. Create an Extension Buildpack to Supply Additional DLLs to a .NET Framework App

An extension buildpack provides the dependencies required by your app that are missing in HWC buildpack. The extension buildpack references and supplies needed dependencies to the app’s container when you push the app.

To create an extension buildpack containing additional DLLs, follow the steps below:

1. Create a list of the dependencies that your published app requires.
2. Copy the DLLs for the dependencies into a ZIP file.
3. On the command line, browse to the directory containing the new ZIP file.
4. To generate a SHA for your ZIP file, run the following command:

```bash
shasum -a256 DLL-ZIP-FILE
```

Where `DLL-ZIP-FILE` is the name of your DLL ZIP file.

5. Upload the ZIP file containing your dependencies to a private web server that is accessible by Cloud Foundry. You can also use an S3 bucket or Azure Storage.

6. On the command line, browse to the directory created by the `buildpack-packager init` command above.
7. Edit the `manifest.yml` file by adding a `dependency` section that references the DLL ZIP file as follows:

```yaml
dependencies:
  - name: BINARY-NAME
    url: http://s3.amazonaws.com/BUCKET-NAME/DLL-ZIP-FILE
    version: 0.0.1
    sha256: GENERATED-SHA-256
cf_stacks:
  - windows
```

Where:
- **BINARY-NAME** is the name of your binary application.
- **BUCKET-NAME** is your S3 bucket name.
- **DLL-ZIP-FILE** is the name of your DLL ZIP file.
- **GENERATED-SHA-256** is the generated SHA.

8. Ensure the `include_files` section of the manifest contains `bin/supply.exe`.

9. To navigate to the BUILDPACK-DIRECTORY/src/BUILDPACK-NAME/ directory, use either Windows Explorer, or your text editor’s ‘File - Open’ option.

10. Edit the `supply.go` script.

11. Add dependencies to the container’s `PATH` as follows:

a. Add a `stager.InstallDependency` for the `dependency` and `version` you have in your manifest.yml file.

b. Add a `stager.AddBinDependency` for every DLL file you want your app to be able to access.

For example:

```go
dep := libbuildpack.Dependency{Name: "my-binary", Version: "0.0.1"}
if err := sInstaller.InstallDependency(dep, s.Stager.Dir()); err != nil {
    return err
}
    return err
}
```

12. To configure the build.sh script to cross-compile, edit `scripts/build.sh` to include the following:

```bash
GOOS=windows go build -ldflags="-s -w" -o bin/supply.exe mysql/supply/cli
```
Step 3. Build the Extension Buildpack

The following steps build a cached extension buildpack artifact.

1. To build a cached buildpack artifact, run the following command:

    ```
    buildpack-packager build-cached -stack STACK-NAME
    ```

    Where:
    
    - **STACK-NAME** is the pre-built operating system that can run apps.

    **Note:** If the buildpack's `supply.go` script contains a mistake, `buildpack-packager` throws an error. You should not push your app with the new extension `buildpack` until after all local errors have been corrected. You must push your app to determine whether the new extension buildpack functions as intended.

Step 4. Validate the Extension Buildpack

The following steps push an app with the cached extension buildpack artifact.

1. Upload the buildpack to a private web server accessible to Cloud Foundry. You can upload the finished buildpack to the same web server as your dependencies. You can also use an S3 bucket or Azure Storage.

2. To deploy your app with the extension buildpack and determine whether the new buildpack functions as intended, run the following command:

    ```
    cf push APP-NAME -s windows -b EXT-BUILDPACK-URL -b hwc_buildpack
    ```

    Where:
    
    - **APP-NAME** is your app’s name.
    - **EXT-BUILDPACK-URL** is the URL of the uploaded extension buildpack zip file.

    For example:

    ```
    $ cf push my-app -s windows -b http://s3.amazon.com/my-bucket/my-example-buildpack.zip -b hwc_buildpack
    ```

    If your extension buildpack does not include the correct dependencies, you will receive an error message.

Step 5. Combine Extension Buildpack with Other Features

If your extension buildpack has executables or scripts that need to be run, you can reference them either in the start command or in profile scripts.

- For more information about the start command, see [command](#) in Deploying with Application Manifests.
- For more information about profile scripts, see [Configure Pre-Runtime Hooks](#) in Deploying an Application.
Tips for .NET Developers

This topic describes how to push .NET apps to Pivotal Application Service (PAS) for Windows cells.

For how to develop .NET microservices for .NET apps using Steeltoe, see the Steeltoe documentation.

Prerequisites

The PAS for Windows tile must be installed and configured. For documentation about installing the PAS for Windows tile, see the Installing and Configuring PAS for Windows topic.

Operators must also install the Cloud Foundry Command Line Interface (cf CLI) to run the commands on this topic. For information about installing the cf CLI, see the Installing the cf CLI topic.

Overview

After operators install and configure the PAS for Windows tile, developers can push .NET apps to the Windows cell. Developers can push both OWIN and non-OWIN apps, and can push apps that are served by Hostable Web Core or self-hosted.

If you have upgraded to PAS for Windows and have apps that you want to migrate, see the Upgrading Cells topic.

Develop .NET Apps

.NET on PCF Cookbook

The .NET Cookbook has useful tips and recipes for developing .NET apps to run on PAS for Windows.

Push a .NET App

By default, PCF serves .NET apps with Hostable Web Core (HWC). HWC is a lighter version of the Internet Information Services (IIS) server that contains the core IIS functionality.

Perform the following steps to push a .NET app to a Windows cell:

1. To target the Cloud Controller of your PCF deployment, run the following command:

   ```
   cf api api.APP-DOMAIN
   ```

   Where `APP-DOMAIN` is your application’s public domain name, for example: `example.com`.

2. Run one of the following commands to deploy your .NET app:

   - To deploy an app with `.bat` or `.exe` files, run the following command:

     ```
     cf push -s windows -b binary_buildpack
     ```

   - To deploy a .NET Framework app, run the following command:

     ```
     cf push APP-NAME -s windows -b hwc_buildpack
     ```

   Where `APP-NAME` is the name of your application.

   **Note:** The `-s windows` option instructs PCF to run the app in the Windows cell.
3. Wait for your app to stage and start. If you see an error message, see the Troubleshoot App Errors section of this topic.

Context Path Routing Support for ASP.NET Apps

Context path routing enables multiple apps to share the same route hostname, such as `app1.example.com/app2`. ASP.NET developers can host apps under a route path. Within Windows cells, you can have multiple routes to an app, but those routes cannot have different context paths.

Making an app accessible under another app’s URL requires a pair of commands. To define a context path route, such as `app1.example.com/app2`, run the following commands:

1. To push the top-level app, run the following command:
   
   ```
   cf push TOP-LEVEL-APP-NAME
   ```
   
   Where `TOP-LEVEL-APP-NAME` is the name of your top-level app.

2. To push the lower-level app, run the following command:
   
   ```
   cf push LOWER-LEVEL-APP-NAME -d APP-DOMAIN --hostname TOP-LEVEL-APP-NAME --route-path LOWER-LEVEL-APP-NAME
   ```
   
   Where:
   - `TOP-LEVEL-APP-NAME` is the name of your top-level app.
   - `LOWER-LEVEL-APP-NAME` is the name of your lower-level app.
   - `APP-DOMAIN` is your application’s public domain name, for example: `example.com`.

   **Note:** The `-d` parameter is only needed when pushing an app to a non-default domain.

Push a Self-Hosted App

Developers can choose to push a self-hosted app instead of using Hostable Web Core. Self-hosted apps combine server code with the app code.

Perform the following steps to push a self-hosted app:

1. To target the Cloud Controller of your PCF deployment, run the following command:
   
   ```
   cf api api.APP-DOMAIN
   ```
   
   Where `APP-DOMAIN` is your application’s public domain name, for example: `example.com`.

2. To push your .NET app from the app root, run the following command:
   
   ```
   cf push APP-NAME -a windows -b binary_buildpack -c PATH-TO-BINARY
   ```
   
   Where:
   - `APP-NAME` is the name of your app.
   - `PATH-TO-BINARY` is the path to your executable.

3. Wait for your app to stage and start. If you see an error message, see the Troubleshoot App Errors section of this topic.

Push a SOAP Service

Developers can push Simple Object Access Protocol (SOAP) web services to their PCF deployment by following the procedures in the sections below.
Step 1: Deploy Your Web Service

Perform the following steps to deploy a SOAP web service:

1. Develop the service as an ASMX web service in Microsoft Visual Studio.
2. Publish the service to your local file system.
3. Open a command line to the directory containing the published web service.
4. To push your service, run the following command:
   
   ```
   cf push SOAP-SERVICE-NAME -s windows -b hwc_buildpack -p WEB-SERVICE-DIRECTORY -u none
   ```
   
   Where
   
   - **SOAP-SERVICE-NAME** is the name of your service.
   - **WEB-SERVICE-DIRECTORY** is the path to the directory containing the published web service.

   For example:
   
   ```
   cf push webservice -s windows -b hwc_buildpack -u none
   ```
   
   - **Note:** The push command must include the `-s` flag to specify the stack, which instructs PCF to run the app in the Windows cell.
   
   - **Note:** The `-p` and `-u` parameters are optional parameters. The `-p` parameter is needed only when pushing your service from a directory that does not contain the published web service. The `-u` parameter is needed only when disabling the health check when you do not have a route serving.

5. Confirm your service is running by finding your service’s URL in the push command’s output and browsing to it. In the example above, the URL for the service would be: [http://webservice.example.com](http://webservice.example.com)

Step 2: Modify the WSDL File

Your SOAP web service is now deployed on PCF, but the service’s WSDL file contains incorrect port information. The WSDL file must be modified to enable an app to consume your web service. Either you or the service developer can perform the needed modification.

Examine the following portion of an example WSDL file:

```
- <wsdl:service name="WebService1">
  - <wsdl:port name="WebService1Soap" binding="tns:WebService1Soap">
    <soap:address location="http://webservice.example.com:62492/WebService1.asmx"/>
  </wsdl:port>
  - <wsdl:port name="WebService1Soap12" binding="tns:WebService1Soap12">
    <soap12:address location="http://webservice.example.com:62492/WebService1.asmx"/>
  </wsdl:port>
- </wsdl:service>
```

The WSDL file provides the port number for the SOAP web service as **62492**. This is the port that the web service listens on in the [Garden container](https://garden-container), but external applications cannot access the service on this port. Instead, external applications must use port **80**, and the [Gorouter](https://gorouter) routes requests to the web service in the container.

The URL of the web service in the WSDL file must be modified to remove **62492**. With no port number, the URL defaults to port **80**. In the example above, the modified URL would be:

[http://webservice.example.com/WebService1.asmx](http://webservice.example.com/WebService1.asmx)

SOAP web service developers can resolve this problem in one of two ways:

- Modify the WSDL file by following the instructions in [Modify a Web Service’s WSDL Using a SoapExtensionReflector](https://microsoft.com) from the Microsoft Developers
Instruct the developers of external applications that consume the web service to perform the steps in the Consume the SOAP Web Service section of this topic.

Consume the SOAP Web Service

Developers of external applications that consume the SOAP web service can perform the following steps to use a modified version of the WSDL file:

1. In a browser, navigate to the WSDL file of the web service, using the following URL:


   Where:
   - SOAP-SERVICE-NAME is the name of your service.
   - APP-DOMAIN is your site’s public domain name.
   - ASMX-FILE is the filename of your asmx file.

   For example:

   https://webservice.example.com/WebService1.asmx?wsdl

2. Download the WSDL file to your local machine.

3. Edit the WSDL file to eliminate the container port, as described in the Modify the WSDL File section of this topic.

4. In Microsoft Visual Studio, right-click on your application in the Solution Explorer and select Add > Service Reference.

5. Under Address, enter the local path to the modified WSDL file.

   For example:

   C:\Users\example\wsdl.xml

6. Click OK. Microsoft Visual Studio generates a client from the WSDL file that you can use in your codebase.

Context Path Routing Support for SOAP Web Services

Developers can push SOAP web services to their PCF deployment with context path routing. For more information, see the Context Path Routing Support for ASP.NET Apps section.

Troubleshoot App Errors

If a .NET app fails to start, consult the following list of errors and their possible solutions:

- **NoCompatibleCell**: Your PCF deployment cannot connect to your Windows cell. For information about troubleshooting your Windows cell configuration, see the Troubleshooting Windows Cells topic.
- **Start unsuccessful**: Your app may be misconfigured or lacks the required DLL files and dependencies.
  - Ensure that your app directory contains either a valid .exe binary or a valid Web.config file.
  - Ensure that you are pushing from a directory containing your app dependencies. If it does not, specify the app dependency directory with the -p flag.
Java Buildpack

You can use the Java buildpack with apps written in Grails, Play, Spring, or any other JVM-based language or framework.

See the following topics for more information:

- Tips for Java Developers
- Getting Started Deploying Apps
- Configuring Service Connections
- Cloud Foundry Java Client Library

See the Java Buildpack Release Notes for information about specific versions. You can find the source for the Java buildpack in the Java buildpack repository on GitHub:

Buildpack and Application Logging

The Java buildpack only runs during the staging process, and therefore only logs staging information such as the downloaded components, configuration data, and work performed on your application by the buildpack.

The Java buildpack source documentation states the following:

- The Java buildpack logs all messages, regardless of severity, to APP-DIRECTORY/.java-buildpack.log. The buildpack also logs messages to Stderr, filtered by a configured severity level.
- If the buildpack fails with an exception, the exception message is logged with a log level of ERROR. The exception stack trace is logged with a log level of DEBUG. This prevents users from seeing stack traces by default.

Once staging completes, the buildpack stops logging. The Loggregator handles application logging.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. For more information, see the Application Logging in Cloud Foundry topic.

BOSH Custom Trusted Certificate Support

Versions 3.7 and later of the Java buildpack support BOSH-configured custom trusted certificates. For more information, see Configuring Trusted Certificates in the BOSH documentation.

The Java buildpack pulls the contents of /etc/ssl/certs/ca-certificates.crt and SCF_INSTANCE_CERT/SCF_INSTANCE_KEY by default.

The log output for Diego Instance Identity-based KeyStore appears as follows:

```
Adding System Key Manager
Adding Key Manager for /etc/cf-instance-credentials/instance.key and /etc/cf-instance-credentials/instance.crt
Start watching /etc/cf-instance-credentials/instance.crt
Start watching /etc/cf-instance-credentials/instance.key
Initialized KeyManager for /etc/cf-instance-credentials/instance.key and /etc/cf-instance-credentials/instance.crt
```

The log output for Diego Trusted Certificate-based TrustStore appears as follows:

```
Adding System Trust Manager
Adding TrustManager for /etc/ssl/certs/ca-certificates.crt
Start watching /etc/ssl/certs/ca-certificates.crt
Initialized TrustManager for /etc/ssl/certs/ca-certificates.crt
```

Memory Constraints in Java Buildpack 4.0

The memory calculator in Java buildpack 4.0 accounts for the following memory regions:

- -Xms: Heap
- -XX:MaxMetaspaceSize: Metaspace
-xx:Thread Stacks

-xx:MaxDirectMemorySize : Direct Memory

-xx:ReservedCodeCacheSize : Code Cache

-xx:CompressedClassSpaceSize : Compressed Class Space

Applications which previously ran in 512 MB or smaller containers may no longer be able to. Most applications will run if they use the Cloud Foundry default container size of 1 G without any modifications. However, you can configure those memory regions directly as needed.

The Java buildpack optimizes for all non-heap memory regions first and leaves the remainder for the heap.

The Java buildpack prints a histogram of the heap to the logs when the JVM encounters a terminal failure.

The Cloud Foundry default Java buildpack is currently 3.x to allows time for apps to be upgrade to 4.x.

For more information, see Java buildpack 4.0.
Cloud Foundry can deploy a number of different JVM-based artifact types. For a more detailed explanation of what it supports, see the Java Buildpack documentation.

Java Buildpack

For detailed information about using, configuring, and extending the Cloud Foundry Java buildpack, see Java Buildpack documentation.

Design

The Java Buildpack is designed to convert artifacts that run on the JVM into executable applications. It does this by identifying one of the supported artifact types (Grails, Groovy, Java, Play Framework, Spring Boot, and Servlet) and downloading all additional dependencies needed to run. The collection of services bound to the application is also analyzed and any dependencies related to those services are also downloaded.

As an example, pushing a WAR file that is bound to a PostgreSQL database and New Relic for performance monitoring would result in the following:

```
Initialized empty Git repository in /tmp/buildpacks/java-buildpack/.git/
-->
Java Buildpack source: https://github.com/cloudfoundry/java-buildpack#0928916a2dd78e9f09469c558046ed9b98015
-->
Downloading Open Jdk JRE 1.7.0_51 from http://.../openjdk/lucid/x86_64/openjdk-1.7.0_51.tar.gz (0.0s)
Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.9s)
-->
Downloading New Relic Agent 3.4.1 from http://.../new-relic/new-relic-3.4.1.jar (0.4s)
-->
Downloading Postgresql JDBC 9.3.1100 from http://.../postgresql-jdbc/postgresql-jdbc-9.3.1100.jar (0.0s)
-->
Downloading Spring Auto Reconfiguration 0.8.7 from http://.../auto-reconfiguration/auto-reconfiguration-0.8.7.jar (0.0s)
Modifying /WEB-INF/web.xml for Auto Reconfiguration
-->
Downloading Tomcat 7.0.50 from http://.../tomcat/tomcat-7.0.50.tar.gz (0.0s)
Expanding Tomcat to .java-buildpack/tomcat (0.1s)
-->
Downloading Buildpack Tomcat Support 1.1.1 from http://.../tomcat-buildpack-support/tomcat-buildpack-support-1.1.1.jar (0.1s)
-->
Uploading droplet (57M)
```

Configuration

In most cases, the buildpack should work without any configuration. If you are new to Cloud Foundry, we recommend that you make your first attempts without modifying the buildpack configuration. If the buildpack requires some configuration, use a fork of the buildpack.

Java Client Library

The Cloud Foundry Client Library provides a Java API for interacting with a Cloud Foundry instance. This library, `cloudfoundry-client-lib`, is used by the Cloud Foundry Maven plugin, the Cloud Foundry Gradle plugin, and other Java-based tools.

For information about using this library, see the Java Cloud Foundry Library page.

Grails

Grails packages applications into WAR files for deployment into a Servlet container. To build the WAR file and deploy it, run the following:

```
$ grails prod war
$ cf push my-application -p target/my-application-version.war
```
Groovy applications based on both Ratpack and a simple collection of files are supported.

**Ratpack**

Ratpack packages applications into two different styles; Cloud Foundry supports the `distZip` style. To build the ZIP and deploy it, run the following:

```
gradle distZip
cf push my-application -p build/distributions/my-application.zip
```

**Raw Groovy**

Groovy applications that are made up of a single entry point plus any supporting files can be run without any other work. To deploy them, run the following:

```
cf push my-application
```

**Java Main**

Java applications with a `main()` method can be run provided that they are packaged as self-executable JARs.

```
Note: If your application is not web-enabled, you must suppress route creation to avoid a “failed to start accepting connections” error. To suppress route creation, add `no-route: true` to the application manifest or use the `--no-route` flag with the `cf push` command.

For more information about the `no-route` attribute, see the Deploying with Application Manifests topic.
```

**Maven**

A Maven build can create a self-executable JAR. To build and deploy the JAR, run the following:

```
mvn package
cf push my-application -p target/my-application-version.jar
```

**Gradle**

A Gradle build can create a self-executable JAR. To build and deploy the JAR, run the following:

```
gradle build
cf push my-application -p build/libs/my-application-version.jar
```

**Play Framework**

The Play Framework packages applications into two different styles. Cloud Foundry supports both the `staged` and `dist` styles. To build the `dist` style and deploy it, run the following:

```
play dist
cf push my-application -p target/universal/my-application-version.zip
```
Spring Boot CLI

Spring Boot CLI can run applications comprised entirely of POGOs. To deploy them, run the following:

```
spring grab *.groovy
cf push my-application
```

Servlet

Java applications can be packaged as Servlet applications.

Maven

A Maven build can create a Servlet WAR. To build and deploy the WAR, run the following:

```
mvn package
cf push my-application -p target/my-application-version.war
```

Gradle

A Gradle build can create a Servlet WAR. To build and deploy the JAR, run the following:

```
gradle build
cf push my-application -p build/libs/my-application-version.war
```

Binding to Services

Information about binding apps to services can be found on the following pages:

- [Service Bindings for Grails Applications](#)
- [Service Bindings for Play Framework Applications](#)
- [Service Bindings for Spring Applications](#)

Java and Grails Best Practices

Provide JDBC driver

The Java buildpack does not bundle a JDBC driver with your application. If your application will access a SQL RDBMS, include the appropriate driver in your application.

Allocate Sufficient Memory

If you do not allocate sufficient memory to a Java application when you deploy it, it may fail to start, or PAS may terminate it. You must allocate enough memory to allow for the following:

- Java heap
- Metaspace, if using Java 8
- PermGen, if using Java 7 or earlier
- Stack size per Thread
JVM overhead

The `config/openjdk_jre.yml` file of the Cloud Foundry Java buildpack contains default memory size and weighting settings for the JRE. See the Open JDK JRE README on GitHub for an explanation of JRE memory sizes and weightings and how the Java buildpack calculates and allocates memory to the JRE for your app.

To configure memory-related JRE options for your app, you either create a custom buildpack and specify this buildpack in your deployment manifest or you override the default memory settings of your buildpack as described in the https://github.com/cloudfoundry/java-buildpack#configuration-and-extension with the properties listed in the Open JDK JRE README. For more information about configuring custom buildpacks and manifests, refer to the Custom Buildpacks and Deploying with Application Manifests topics.

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Troubleshoot Out Of Memory

A Java app may crash because of insufficient memory on the Garden container or the JVM on which it runs. See the following sections for help diagnosing and resolving such issues.

JVM

- **Error:** `java.lang.OutOfMemoryError`. See the following example:

  ```
  cf logs APP-NAME --recent
  2016-06-20T09:18:51.00+0100 [APP/0] OUT java.lang.OutOfMemoryError: Metaspace
  ```

- **Cause:** If the JVM cannot garbage-collect enough space to ensure the allocation of a data-structure, it fails with `java.lang.OutOfMemoryError`. In the example above, JVM has an under-sized metaspace. You may see failures in other memory pools, such as heap.

- **Solution:** Configure the JVM correctly for your app. See Allocate Sufficient Memory.

Garden Container

- **Note:** The solutions in this section require configuring the memory calculator, which is a sub-project of the CF Java buildpack that calculates suitable memory settings for Java apps when you push them. See the java-buildpack-memory-calculator repository for more information. If you have questions about the memory calculator, you can ask them in the #java-buildpack channel of the Cloud Foundry Slack organization.

  ```
  cf events APP-NAME
  time     event    actor  description
  2016-06-20T09:18:51.00+0100  app.crash  app-name  index: 0, reason: CRASHED, exit_description: out of memory, exit_status: 255
  ```

This error appears when the JVM allocates more OS-level memory than the quota requested by the app, such as through the manifest.

- **Cause 1 - Insufficient native memory:** This error commonly means that the JVM requires more native memory. In the scope of the Java buildpack and the memory calculator, the term native means the memory required for the JVM to work, along with forms of memory not covered in the other classifications of the memory calculator. This includes the memory footprint of OS-level threads, direct NIO buffers, code cache, program counters, and others.

- **Solution 1:** Determine how much native memory a Java app needs by measuring it with realistic workloads and fine-tuning it accordingly. You can then configure the Java buildpack using the native setting of the memory calculator, as in the example below:

  ```yaml
  applications:
  - name: app-name
    memory: 1G
    env:
      JBP_CONFIG_OPEN_JDK_JRE: {memory_calculator: {memory_sizes: {native: 150m}}}
  ```

  This example shows that 150m of the overall available 1G is reserved for anything that is not heap, metaspace, or permgen. In less common cases, this may come from companion processes started by the JVM, such as the Process API.

- **Cause 2 - High thread count:** Java threads in the JVM can cause memory errors at the Garden level. When an app is under heavy load, it uses a high number of threads and consumes a significant amount of memory.
• **Solution 2:** Set the reserved memory for stack traces to the correct value for your app.

You can use the `stack` setting of the memory calculator to configure the amount of space the JVM reserves for each Java thread. You must multiply this value by the number of threads your app requires. Specify the number of threads in the `stack_threads` setting of the memory calculator. For example, if you estimate the max thread count for an app at 800 and the amount of memory needed to represent the deepest stacktrace of a Java thread is 512KB, configure the memory calculator as follows:

```yaml
---
applications:
- name: app-name
  memory: 1G
  env:
    JBP_CONFIG_OPEN_JDK_JRE: [
      memory_calculator:
        {stack_threads: 800, memory_sizes: {stack: 512k}}]
---
```

In this example, the overall memory amount reserved by the JVM for representing the stacks of Java threads is \(800 \times 512k = 400m\).

The correct settings for `stack` and `stack_threads` depend on your app code, including the libraries it uses. Your app may technically have no upper limit, such as in the case of cavalier usage of `CachedThreadPool` executors. However, you still must calculate the depth of the thread stacks and the amount of space the JVM should reserve for each of them.

### Troubleshoot Failed Upload

If your application fails to upload when you push it to Cloud Foundry, it may be for one of the following reasons:

- **WAR is too large:** An upload may fail due to the size of the WAR file. Cloud Foundry testing indicates WAR files as large as 250 MB upload successfully. If a WAR file is larger than that fails to upload, it may be a result of the file size.

- **Connection issues:** Application uploads can fail if you have a slow Internet connection, or if you upload from a location that is very remote from the target Cloud Foundry instance. If an application upload takes a long time, your authorization token can expire before the upload completes. A workaround is to copy the WAR to a server that is closer to the Cloud Foundry instance, and push it from there.

- **Out-of-date cf CLI client:** Upload of a large WAR is faster and hence less likely to fail if you are using a recent version of the cf CLI. If you are using an older version of the cf CLI client to upload a large WAR, and having problems, try updating to the latest version of the cf CLI.

- **Incorrect WAR targeting:** By default, `cf push` uploads everything in the current directory. For a Java application, `cf push` with no option flags uploads source code and other unnecessary files, in addition to the WAR. When you push a Java application, specify the path to the WAR:

  ```bash
  cf push MY-APP -p PATH/TO/WAR-FILE
  ```

You can determine whether or not the path was specified for a previously pushed application by examining the application deployment manifest, `manifest.yml`. If the `path` attribute specifies the current directory, the manifest includes a line like the following:

```yaml
path: .
```

To re-push just the WAR, do one of the following:

- Delete `manifest.yml` and run `cf push` again, specifying the location of the WAR using the `-p` flag.
- Edit the `path` argument in `manifest.yml` to point to the WAR, and re-push the application.

### Debug Java Apps on Cloud Foundry

Because of the way that Cloud Foundry deploys your applications and isolates them, it is not possible to connect to your application with the remote Java debugger. Instead, instruct the application to connect to the Java debugger on your local machine.

Here are the instructions for setting up remote debugging when using BOSH Lite or a Cloud Foundry installation.

1. Open your project in Eclipse.
2. Right-click on your project, go to **Debug** as and pick **Debug Configurations**.
3. Create a new **Remote Java Application**.
4. Make sure your project is selected, pick **Standard (Socket Listen)** from the **Connection Type** drop down and set a port. Make sure this port is open if you are running a firewall.
5. Click **Debug**.
The debugger should now be running. If you switch to the Debug perspective, you should see your application listed in the Debug panel and it should say Waiting for vm to connect at port.

Next, push your application to Cloud Foundry and instruct Cloud Foundry to connect to the debugger running on your local machine using the following instructions:

1. Edit your `manifest.yml` file. Set the instances count to 1. If you set this greater than one, multiple applications try to connect to your debugger.

2. Also in `manifest.yml`, add an `env` block and create a variable named `JAVA_OPTS`. For more information about the `env` block, see Environment Variables in Deploying with Application Manifests.

3. Add the remote debugger configuration to the `JAVA_OPTS` variable:
   ```
   -agentlib:jdwp=transport=dt_socket,address=YOUR-IP-ADDRESS:YOUR-PORT
   ```

4. Save the `manifest.yml` file.

5. Run `cf push`.

Upon completion, you should see that your application has started and is now connected to the debugger running in your IDE. You can now add breakpoints and interrogate the application just as you would if it were running locally.

### Slow Starting Java or Grails Apps

Some Java and Grails applications do not start quickly, and the health check for an application can fail if an application starts too slowly.

The current Java buildpack implementation sets the Tomcat `bindOnInit` property to `false`. This prevents Tomcat from listening for HTTP requests until an application has fully deployed.

If your application does not start quickly, the health check may fail because it checks the health of the application before the application can accept requests. By default, the health check fails after a timeout threshold of 60 seconds.

To resolve this issue, use `cf push APP-NAME` with the `-t TIMEOUT-THRESHOLD` option to increase the timeout threshold. Specify `TIMEOUT-THRESHOLD` in seconds.

```
$ cf push my-app -t 180
```

**Note:** The timeout threshold cannot exceed 180 seconds. Specifying a timeout threshold greater than 180 seconds results in the following error:

```
Server error, status code: 400, error code: 100001, message: The app is invalid: health_check_timeout maximum_exceeded
```

### Extension

The Java Buildpack is also designed to be easily extended. It creates abstractions for three types of components (containers, frameworks, and JREs) in order to allow users to easily add functionality. In addition to these abstractions, there are a number of utility classes for simplifying typical buildpack behaviors.

As an example, the New Relic framework looks like the following:
class NewRelicAgent < JavaBuildpack::Component::VersionedDependencyComponent

# @macro base_component_compile
def compile
  FileUtils.mkdir_p logs_dir

  download_jar
  @droplet.copy_resources
end

# @macro base_component_release
def release
  @droplet.java_opts
    .add_javaagent(@droplet.sandbox + jar_name)
    .add_system_property(newrelic.home, @droplet.sandbox)
    .add_system_property(newrelic.config.license_key, license_key)
    .add_system_property(newrelic.config.app_name, "#{application_name}")
    .add_system_property(newrelic.config.log_file_path, logs_dir)
end

protected

# @macro versioned_dependency_component_supports
def supports?
  @application.services.one_service? FILTER, 'licenseKey'
end

private

FILTER = /newrelic\freeze

def application_name
  @application.details['application_name']
end

def license_key
  @application.services.find_service(FILTER['credentials']['licenseKey'])
end

def logs_dir
  @droplet.sandbox + 'logs'
end
end

Environment Variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` as follows:

```
System.getenv("VCAP_SERVICES");
```

See the Cloud Foundry Environment Variables topic for more information.
Getting Started Deploying Java Apps

This topic provides links to additional information about getting started deploying apps using the Java buildpack. See the following topics for deployment guides specific to your app framework:

- Grails
- Ratpack
- Spring
Getting Started Deploying Grails Apps

This guide is intended to walk you through deploying a Grails app to Pivotal Application Service (PAS). If you experience a problem following the steps below, refer to the Known Issues or Troubleshooting Application Deployment and Health topics for more information.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_grails.git` to clone the `pong_matcher_grails` app from GitHub, and follow the instructions in the Sample App Step sections.

Note: Ensure that your Grails app runs locally before continuing with this procedure.

Deploy a Grails Application

This section describes how to deploy a Grails application to PAS.

Prerequisites

- A Grails app that runs locally on your workstation
- Intermediate to advanced Grails knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

Note: You can develop Grails applications in Groovy, Java 7 or 8, or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7 as described in Step 8: Configure the Deployment Manifest of this topic.

Step 1: Declare App Dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle, Grails, and Maven, and provides documentation links for each build tool.

<table>
<thead>
<tr>
<th>Build Tool</th>
<th>Build Script</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradle</td>
<td>build.gradle</td>
<td>Gradle User Guide</td>
</tr>
<tr>
<td>Grails</td>
<td>BuildConfig.groovy</td>
<td>Grails: Configuration - Reference Documentation</td>
</tr>
<tr>
<td>Maven</td>
<td>pom.xml</td>
<td>Apache Maven Project Documentation</td>
</tr>
</tbody>
</table>

Sample App Step
You can skip this step. The `pong_matcher_grails/app/grails-app/conf/BuildConfig.groovy` file contains the dependencies for the `pong_matcher_grails` sample app, as the example below shows.

```groovy
dependencies {
    // specify dependencies here under either 'build', 'compile', 'runtime', 'test' or 'provided' scopes e.g.
    // runtime 'mysql:mysql-connector-java:5.1.29'
    // runtime 'org.postgresql:postgresql:9.3-1101-jdbc41'
    test "org.grails:grails-datastore-test-support:1.0-grails-2.4"
    runtime 'mysql:mysql-connector-java:5.1.33'
}
```
Step 2: Allocate Sufficient Memory

Run the Cloud Foundry Command Line Interface (cf CLI) `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of 64M, 128M, 256M, 512M, 1G, or 2G. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

```
dataSource {
  pooled = true
  jmxExport = true
  driverClassName = "com.mysql.jdbc.Driver"
  dialect = org.hibernate.dialect.MySQL5InnoDBDialect
  url = new URI(System.env.DATABASE_URL ?: "mysql://foo:bar@localhost")
  username = uri.userInfo ? uri.userInfo.split(':')[0] : ""
  password = uri.userInfo ? uri.userInfo.split(':')[1] : ""
  properties {
    dbProperties {
      autoReconnect = true
    }
  }
}
```

Step 4: (Optional) Configure a Procfile

Use a Procfile to declare required runtime processes for your web app and to specify your web server. For more information, see the Configuring a Production Server topic.

```
Sample App Step
You can skip this step. The `pong_matcher_grails` sample app does not require a Procfile.
```

Step 5: Create and Bind a Service Instance for a Grails Application

This section describes using the cf CLI to configure a ClearDB managed service instance for an app. You can use either the CLI or Apps Manager to perform this task.
PAS supports two types of service instances:

- Managed services integrate with PAS through service brokers that offer services and plans and manage the service calls between PAS and a service provider.

- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

Run the `cf CLI ` command to view managed and user-provided services and plans available to you.

The example shows two of the available managed database-as-a-service providers and their offered plans: `cleardb` database-as-a-service for MySQL-powered apps and `elephantsql` PostgreSQL as a Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

<table>
<thead>
<tr>
<th>service</th>
<th>plans</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cleardb</td>
<td>spark, boost, amp</td>
<td>Highly available MySQL for your apps</td>
</tr>
<tr>
<td>elephantsql</td>
<td>turtle, panda, elephant</td>
<td>PostgreSQL as a Service</td>
</tr>
</tbody>
</table>
```

Run `cf create-service SERVICE PLAN SERVICE-INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE-INSTANCE.

```
$ cf create-service cleardb spark mysql
Creating service mysql in org Cloud-Apps / space development as clouduser@example.com.... OK
```

Bind a Service Instance

When you bind an app to a service instance, PAS writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider’s documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE-INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using the Apps Manager.

```
Sample App Step
You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.
```

---

applications:
services:
- mysql

Step 6: Configure the Deployment Manifest
You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample App Step
You can skip this step. The `manifest.yml` file for the `pong_matcher_grails` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and Target the API Endpoint
Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step
You must do this step to run the sample app.

Step 8: Deploy the Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.war` to deploy your application.

Note: You must deploy the `.war` artifact for a Grails app, and you must include the path to the `.war` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the Grails section in the Tips for Java Developers topic.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step
1. Change to the `app` directory, and run `./grailswar` to build the app.
2. Run `cf push pong_matcher_grails -n HOST-NAME` to push the app.
The example below shows the terminal output of deploying the `pong_matcher_grails` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and follows the instructions in the manifest to start two instances of the app, allocating 1 GB of memory between the instances. After the instances start, the output displays their health and status.

```
$ cf push pong_matcher_grails -n my-grails-app
Using manifest file /Users/example/workspace/pong_matcher_grails/app/manifest.yml
Creating app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com... OK
Creating route my-grails-app.cfapps.io... OK
Binding my-grails-app.cfapps.io to pong_matcher_grails... OK
Uploading pong_matcher_grails...
Uploading app files from: /Users/example/workspace/pong_matcher_grails/app/target/pong_matcher_grails-0.1.war
Uploading 4.8M, 704 files OK
Binding service mysql to app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com... OK
Starting app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com... OK
----- Downloaded app package (38M)
----- Java Buildpack Version: v2.5 | https://github.com/cloudfoundry/java-buildpack.git#840500e
----- Downloading Open Jdk JRE 18.0.25 from https://download.run.pivotal.io/openjdk/jre/18.0.25.tar.gz (1.5s)
Expanding Open Jdk JRE to /java-buildpack/open_jdk_jre (1.1s)
----- Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.5.0_RELEASE.jar (0.0s)
Modifying /WEB-INF/web.xml for Auto Reconfiguration
----- Downloading Tomcat Instance 8.0.14 from https://download.run.pivotal.io/tomcat/tomcat-8.0.14.tar.gz (0.4s)
Expanding Tomcat to /java-buildpack/tomcat (0.1s)
----- Downloading Tomcat Lifecycle Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-lifecycle-support/tomcat-lifecycle-support-2.4.0_RELEASE.jar (0.0s)
----- Downloading Tomcat Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-logging-support/tomcat-logging-support-2.4.0_RELEASE.jar (0.0s)
----- Downloading Tomcat Access Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-access-logging-support/tomcat-access-logging-support-2.4.0_RELEASE.jar (0.0s)
----- Uploading droplet (83M)
0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
2 of 2 instances running
App started

Showing health and status for app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 2/2
usage: 1G x 2 instances
urls: my-grails-app.cfapps.io

<table>
<thead>
<tr>
<th>state</th>
<th>since</th>
<th>cpu</th>
<th>memory</th>
<th>disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 running</td>
<td>2014-11-10 05:07:33 PM</td>
<td>0.0%</td>
<td>686.4M of 1G</td>
<td>153.6M of 1G</td>
</tr>
<tr>
<td>91 running</td>
<td>2014-11-10 05:07:36 PM</td>
<td>0.0%</td>
<td>677.23M of 1G</td>
<td>153.6M of 1G</td>
</tr>
</tbody>
</table>
```

Step 9: Test Your Deployed App

You've deployed an app to PAS!

Use the `cf` CLI or Apps Manager to review information and administer your app and your PAS account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.
Sample App Step
To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:

   ```bash
   curl -X DELETE HOST/all
   ```
   
   You should get a response of 200.

3. To request a match as “andrew”, run:

   ```bash
   curl -X PUT "http://HOST/match_requests/firstrequest" -d '{"player": "andrew"}"
   ```
   
   You should again get a response of 200.

4. To request a match as a different player, run:

   ```bash
   curl -X PUT "http://HOST/match_requests/secondrequest" -d '{"player": "navratilova"}"
   ```

5. To check the status of the first match request, run:

   ```bash
   curl -X GET HOST/match_requests/firstrequest
   ```
   
   The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:

   ```bash
   curl -X POST "http://HOST/results" -d '{"match_id": "MATCH_ID", "winner": "andrew", "loser": "navratilova"}"
   ```
   
   You should receive a 201 Created response.

Manage Your Application with the cf CLI
Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Note: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands:

| error code: 10003, message: You are not authorized to perform the requested action |

For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting
If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails
Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

App Requires a Unique URL
PAS requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:
• --host to assign a different HOST name for the app.

• --random-route to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.
Getting Started Deploying Ratpack Apps

Page last updated:

This guide is intended to walk you through deploying a Ratpack app to Pivotal Application Service (PAS). If you experience a problem following the steps below, check the [Known Issues](#) topic or refer to the [Troubleshooting Application Deployment and Health](#) topic.

**Sample App Step**

If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_groovy.git` to clone the `pong_matcher_groovy` app from GitHub, and follow the instructions in the Sample App Step sections.

**Note:** Ensure that your Ratpack app runs locally before continuing with this procedure.

Deploy a Ratpack Application

This section describes how to deploy a Ratpack application to PAS.

**Prerequisites**

- A Ratpack app that runs locally on your workstation
- Intermediate to advanced Ratpack knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

**Note:** You can develop Ratpack applications in Java 7 or 8 or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7. Refer to Step 8: Configure the Deployment Manifest.

**Step 1: Declare App Dependencies**

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

<table>
<thead>
<tr>
<th>Build Tool</th>
<th>Build Script</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradle</td>
<td><code>build.gradle</code></td>
<td><a href="#">Gradle User Guide</a></td>
</tr>
<tr>
<td>Maven</td>
<td><code>pom.xml</code></td>
<td><a href="#">Apache Maven Project Documentation</a></td>
</tr>
</tbody>
</table>

**Sample App Step**

You can skip this step. The `build.gradle` file contains the dependencies for the `pong_matcher_groovy` sample app, as the example below shows.

```groovy
dependencies {
    // SpringLoaded enables runtime hot reloading.
    // It is not part of the app runtime and is not shipped in the distribution.
    springloaded "org.springframework:springloaded:1.2.0.RELEASE"

    // Default SLF4J binding. Note that this is a blocking implementation.
    // See here for a non blocking appender http://logging.apache.org/log4j/2.x/manual/async.html
    runtime "org.slf4j:slf4j-simple:1.7.7"

    compile group: 'redis.clients', name: 'jedis', version: '2.5.2', transitive: true

    testCompile "org.spockframework:spock-core:0.7-groovy-2.0"
}
```

© Copyright Pivotal Software Inc, 2013-2019

1374
Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of 64M, 128M, 256M, 512M, 1G, or 2G. For example:

```
cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

---

**Sample App Step**

You can skip this step. In the `manifest.yml` of the `pong_matcher_groovy` sample app, the `memory` sub-block of the `applications` block allocates 512 MB to the app.

---

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

---

**Sample App Step**

You can skip this step. The `pong_matcher_groovy` sample app does not require a JDBC driver.

---

Step 4: (Optional) Configure a Procfile

Use a Procfile to declare required runtime processes for your web app and to specify your web server. For more information, see the Configuring a Production Server topic.

---

**Sample App Step**

You can skip this step. The `pong_matcher_groovy` app does not require a Procfile.

---

Step 5: Create and Bind a Service Instance for a Ratpack Application

This section describes using the CLI to configure a Redis managed service instance for an app. You can use either the CLI or Apps Manager to perform this task.

PAS supports two types of service instances:

- Managed services integrate with PAS through service brokers that offer services and plans and manage the service calls between PAS and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans available to you.

The example shows two of the available managed database-as-a-service providers and their offered plans: `elephantsql` PostgreSQL as a Service and `rediscloud` Enterprise-Class Redis for Developers.
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...

OK

<table>
<thead>
<tr>
<th>service</th>
<th>plans</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elephantql</td>
<td>turtle, panda, elephant</td>
<td>PostgreSQL as a Service</td>
</tr>
<tr>
<td>rediscloud</td>
<td>30mb, 100mb, 1gb, 10gb, 50gb</td>
<td>Enterprise-Class Redis for Developers</td>
</tr>
</tbody>
</table>

Run `cf create-service SERVICE PLAN SERVICE-INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE-INSTANCE.

Sample App Step
Run `cf create-service rediscloud 30mb baby-redis` to create a service instance named `baby-redis` that uses the `rediscloud` service and the `30mb` plan, as the example below shows.

OK

Bind a Service Instance
When you bind an app to a service instance, PAS writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider’s documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using the Apps Manager.

Sample App Step
You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```yaml
---
applications:
  ...
  services:
    - baby-redis
```

Step 6: Configure the Deployment Manifest
You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample App Step
You can skip this step. The `manifest.yml` file for the `pong_matcher_groovy` sample app does not require any additional configuration to deploy the app.
Step 7: Log in and Target the API Endpoint

Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step

You must perform this step to run the sample app.

Step 8: Deploy the Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.distZip` to deploy your application.

Note: You must deploy the .distZip artifact for a Ratpack app, and you must include the path to the .distZip file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the Tips for Java Developers topic.

cf push APP-NAME creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is shared-domain.example.com. For example: `cf push my-app` creates the URL my-app.shared-domain.example.com.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

1. Change to the app directory, and run `./gradlew distZip` to build the app.
2. Run `cf push pong_matcher_groovy -n HOST-NAME` to push the app.

Example: `cf push pong_matcher_groovy -n groovy-ratpack-app`

Note: You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the pong_matcher_groovy app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the baby-redis service and follows the instructions in the manifest to start one instance of the app with 512 MB. After the app starts, the output displays the health and status of the app.
Step 9: Test Your Deployed App

You’ve deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

---

Sample App Step
To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:
   ```
curl -X DELETE
HOST/all
```
   You should get a response of 200.

3. To request a match as "andrew", run:
   ```
curl -H "Content-Type: application/json" -X PUT
HOST/match_requests/firstrequest -d '{"player":"andrew"}'
```
   You should again get a response of 200.

4. To request a match as a different player, run:
   ```
curl -H "Content-Type: application/json" -X PUT
HOST/match_requests/secondrequest -d '{"player":"navratilova"}'
```

5. To check the status of the first match request, run:
Manage Your Application with the cf CLI

Run `cf` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

App Requires a Unique URL

PAS requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.
Getting Started Deploying Spring Apps

Page last updated:

This guide is intended to walk you through deploying a Spring app to Pivotal Application Service (PAS). You can choose whether to push a sample app, your own app, or both.

If you experience a problem following the steps below, see the Known Issues topic, or refer to the Troubleshooting Application Deployment and Health topic.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_spring` to clone the pong_matcher_spring app from GitHub, and follow the instructions in the Sample App Step sections.

Note: Ensure that your Spring app runs locally before continuing with this procedure.

Deploy a Spring Application

This section describes how to deploy your Spring application to PAS.

Prerequisites
- A Spring app that runs locally on your workstation
- Intermediate to advanced Spring knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.6, 1.7, or 1.8 for Java 6, 7, or 8 configured on your workstation

Note: The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to an earlier version. For more information, refer to the Custom Buildpacks topic.

Step 1: Declare App Dependencies

Be sure to declare all the dependency tasks for your app in the build script of your chosen build tool.

The Spring Getting Started Guides demonstrate features and functionality you can add to your app, such as consuming RESTful services or integrating data. These guides contain Gradle and Maven build script examples with dependencies. You can copy the code for the dependencies into your build script.

The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

<table>
<thead>
<tr>
<th>Build Tool</th>
<th>Build Script</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradle</td>
<td>build.gradle</td>
<td>Gradle User Guide</td>
</tr>
<tr>
<td>Maven</td>
<td>pom.xml</td>
<td>Apache Maven Project Documentation</td>
</tr>
</tbody>
</table>

Sample App Step
You can skip this step. The pom.xml file contains the dependencies for the pong_matcher_spring sample app, as the example below shows.
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
  </dependency>
  <dependency>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-core</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.jayway.jsonpath</groupId>
    <artifactId>json-path</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

Note: Make sure you are not building fully executable jars because application push may fail.

Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of 64M, 128M, 256M, 512M, 1G, or 2G. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Sample App Step

You can skip this step. The Cloud Foundry Java buildpack uses settings declared in the sample app to allocate 1 GB of memory to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step

You can skip this step. In the `pong_matcher_spring` sample app, the `src/main/resources/application.yml` file declares the JDBC driver, and the `pom.xml` file includes the JDBC driver as a dependency.

Step 4: Configure Service Connections for a Spring App
PAS provides extensive support for creating and binding a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. For more information about creating and binding a service connection for your app, refer to the Configure Service Connections for Spring topic.

Sample App Step: Create a Service Instance
Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `spark` plan, as the example below shows.

```
cf create-service cleardb spark mysql
```

Creating service mysql in org Cloud-Apps / space development as a.user@example.com ...
OK

Sample App Step: Bind a Service Instance
You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
... 
services:
  - mysql
```

Step 5: Configure the Deployment Manifest
You can specify deployment options in a manifest file `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample App Step
You can skip this step. The `manifest.yml` file for the `pong_matcher_spring` sample app does not require any additional configuration to deploy the app.

Step 6: Log in and Target the API Endpoint
Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step
You must do this step to run the sample app.

Step 7: Deploy Your Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.war` to deploy your application.

Note: Most Spring apps include an artifact, such as a `.jar`, `.war`, or `.zip` file. You must include the path to this file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. The example shows how to specify a path to the `.war` file.
The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

The example below shows the terminal output of deploying the `pong_matcher_spring` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and starts one instance of the app with 1 GB of memory. After the app starts, the output displays the health and status of the app.
Step 8: Test Your Deployed App

You've deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you can edit the manifest.yml to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

**Sample App Step**

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:
   ```
curl -v -X DELETE $HOST/all
   ```
   You should get a response of 200.

3. To request a match as "andrew", run:
   ```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player": "andrew"}'
   ```
   You should again get a response of 200.

4. To request a match as a different player, run:
   ```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player": "navratilova"}'
   ```
Manage Your App with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Note: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands:

```
error code: 10003, message: You are not authorized to perform the requested action
```
For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

App Requires a Content-Type

If you specify a `Content-Encoding` header of `gzip` but do not specify a `Content-Type` within your application, PAS might send a `Content-Type` of `application/x-gzip` to the browser. This scenario might cause the deploy to fail if it conflicts with the actual encoded content of your app. To avoid this issue, be sure to explicitly set `Content-Type` within your app.

App Requires a Unique URL

PAS requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-h` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.
Configuring Service Connections

This topic provides links to additional information about configuring service connections for Java apps. See the specific documentation for your app framework:

- Grails
- Play
- Spring
Configuring Service Connections for Grails

Cloud Foundry provides extensive support for connecting a Grails application to services such as MySQL, Postgres, MongoDB, Redis, and RabbitMQ. In many cases, a Grails application running on Cloud Foundry can automatically detect and configure connections to services. For more advanced cases, you can control service connection parameters yourself.

Auto-Configuration

Grails provides plugins for accessing SQL (using Hibernate), MongoDB, and Redis services. If you install any of these plugins and configure them in your Config.groovy or DataSource.groovy file, Cloud Foundry reconfigures the plugin with connection information when your app starts.

If you use all three types of services, your configuration might look like this:

```groovy
environments { 
    production { 
        dataSource { 
            url = 'jdbc:mysql://localhost/db?useUnicode=true&characterEncoding=utf8' 
            dialect = org.hibernate.dialect.MySQLInnoDBDialect 
            driverClassName = 'com.mysql.jdbc.Driver' 
            username = 'user' 
            password = "password" 
        } 
        grails { 
            mongo { 
                host = 'localhost' 
                port = 27017 
                databaseName = 'foo' 
                username = 'user' 
                password = 'password' 
            } 
            redis { 
                host = 'localhost' 
                port = 6379 
                password = 'password' 
                timeout = 2000 
            } 
        } 
    } 
}
```

The `url`, `host`, `port`, `databaseName`, `username`, and `password` fields in this configuration will be overriden by the Cloud Foundry auto-reconfiguration if it detects that the application is running in a Cloud Foundry environment. If you want to test the application locally against your own services, you can put real values in these fields. If the application will only be run against Cloud Foundry services, you can put placeholder values as shown here, but the fields must exist in the configuration.

Manual Configuration

If you do not want to use auto-configuration, you can configure the Cloud Foundry service connections manually.

Follow the steps below to manually configure a service connection.

1. Add the `spring-cloud` library to the `dependencies` section of your BuildConfig.groovy file.

```groovy
repositories { 
    grailsHome() 
    mavenCentral() 
    grailsCentral() 
    mavenRepo "http://repo.spring.io/milestone" 
}

dependencies { 
    compile "org.springframework.cloud:spring-cloud-cloudfoundry-connector:1.0.0.RELEASE" 
    compile "org.springframework.cloud:spring-cloud-spring-service-connector:1.0.0.RELEASE" 
}
```

Adding the `spring-cloud` library allows you to disable auto-configuration and use the `spring-cloud` API in your `DataSource.groovy` file to set the
connection parameters.

2. Add the following to your `grails-app/conf/spring/resources.groovy` file to disable auto-configuration:

```groovy
beans = {
  cloudFactory(org.springframework.cloud.CloudFactory)
}
```

3. Add the following imports to your `DataSource.groovy` file to allow spring-cloud API commands:

```groovy
import org.springframework.cloud.CloudFactory
import org.springframework.cloud.CloudException
```

4. Add the following code to your `DataSource.groovy` file to enable Cloud Foundry's `getCloud` method to function locally or in other environments outside of a cloud.

```groovy
def cloud = null
try {
  cloud = new CloudFactory().cloud
} catch (CloudException) {}
```

5. Use code like the following to access the cloud object:

```groovy
def dbInfo = cloud?.getServiceInfo('myapp-mysql')
url = dbInfo?.jdbcUrl
username = dbInfo?userName
password = dbInfo?password
```

`myapp-mysql` is the name of the service as it appears in the `name` column of the output from `cf services`. For example, `mysql` or `rabbitmq`.

The example `DataSource.groovy` file below contains the following:

- The imports that allow spring-cloud API commands
- The code that enables the `getCloud` method to function locally or in other environments outside of a cloud
- Code to access the cloud object for SQL, MongoDB, and Redis services
import org.springframework.cloud.CloudFactory
import org.springframework.cloud.CloudException

def cloud = null
try {
    cloud = new CloudFactory().cloud
} catch (CloudException) {}
Configuring Service Connections for Play Framework

Page last updated:

Cloud Foundry supports running Play Framework applications and the Play JPA plugin for auto-configuration for Play versions up to and including v2.4.x.

Cloud Foundry provides support for connecting a Play Framework application to services such as MySQL and Postgres. In many cases, a Play Framework application running on Cloud Foundry can automatically detect and configure connections to services.

Auto-Configuration

By default, Cloud Foundry detects service connections in a Play Framework application and configures them to use the credentials provided in the Cloud Foundry environment. Note that auto-configuration happens only if there is a single service of either of the supported types—MySQL or Postgres.
Configuring Service Connections for Spring

Cloud Foundry provides extensive support for connecting a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. In many cases, Cloud Foundry can automatically configure a Spring application without any code changes. For more advanced cases, you can control service connection parameters yourself.

Auto-Reconfiguration

If your Spring application requires services such as a relational database or messaging system, you might be able to deploy your application to Cloud Foundry without changing any code. In this case, Cloud Foundry automatically re-configures the relevant bean definitions to bind them to cloud services.

For information about connecting to services from a Spring application, see Spring Cloud Spring Service Connector.

Cloud Foundry auto-reconfigures applications only if the following items are true for your application:

- Only one service instance of a given service type is bound to the application. In this context, different relational databases services are considered the same service type. For example, if both a MySQL and a PostgreSQL service are bound to the application, auto-reconfiguration does not occur.
- Only one bean of a matching type is in the Spring application context. For example, you can have only one bean of type `javax.sql.DataSource`.

With auto-reconfiguration, Cloud Foundry creates the `DataSource` or connection factory bean itself, using its own values for properties such as host, port, and username. For example, if you have a single `javax.sql.DataSource` bean in your application context that Cloud Foundry auto-reconfigures and binds to its own database service, Cloud Foundry does not use the username, password, and driver URL you originally specified. Instead, Cloud Foundry uses its own internal values. This is transparent to the app, which really only cares about having a `DataSource` where it can write data but does not really care what the specific properties are that created the database. Also, if you have customized the configuration of a service, such as the pool size or connection properties, Cloud Foundry auto-reconfiguration ignores the customizations.

For more information about auto-reconfiguration of specific services types, see the Service-Specific Details section.

Manual Configuration

Use manual configuration if you have multiple services of a given type or you want to have more control over the configuration than auto-reconfiguration provides.

To use manual configuration, include the `spring-cloud` library in your list of application dependencies. Update your application Maven `pom.xml` or Gradle `build.gradle` file to include dependencies on the `org.springframework.cloud:spring-cloud-spring-service-connector` and `org.springframework.cloud:spring-cloud-cloudfoundry-connector` artifacts.

For example, if you use Maven to build your application, the following `pom.xml` snippet shows how to add this dependency.

```xml
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-spring-service-connector</artifactId>
    <version>1.2.3.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-cloudfoundry-connector</artifactId>
    <version>1.2.3.RELEASE</version>
  </dependency>
</dependencies>
```

You also need to update your application build file to include the Spring Framework Milestone repository. The following `pom.xml` snippet shows how to do this for Maven:

```xml
<repositories>
  <repository/>
  <repository/>
</repositories>
```
Java Configuration

Typical use of Java config involves extending the `AbstractCloudConfig` class and adding methods with the `@Bean` annotation to create beans for services. Apps migrating from auto-reconfiguration might first try Scanning for Services until they need more explicit control. Java config also offers a way to expose application and service properties. Use this for debugging or to create service connectors using a lower-level access.

Create a Service Bean

In the following example, the configuration creates a `DataSource` bean connecting to the only relational database service bound to the app. It also creates a `MongoDbFactory` bean, again, connecting to the only MongoDB service bound to the app. Check Javadoc for `AbstractCloudConfig` for ways to connect to other services.

```java
class CloudConfig extends AbstractCloudConfig {

    @Bean
    public DataSource inventoryDataSource() {
        return connectionFactory().dataSource();
    }

    // ... more beans to obtain service connectors
}
```

The bean names match the method names unless you specify an explicit value to the annotation such as `@Bean("inventory-service")`, following Spring’s Java configuration standards.

If you have more than one service of a type bound to the app or want to have an explicit control over the services to which a bean is bound, you can pass the service names to methods such as `dataSource()` and `mongoDbFactory()` as follows:

```java
class CloudConfig extends AbstractCloudConfig {

    @Bean
    public DataSource inventoryDataSource() {
        return connectionFactory("inventory-db-service").dataSource();
    }

    @Bean
    public MongoDbFactory documentMongoDbFactory() {
        return connectionFactory("document-service").mongoDbFactory();
    }

    // ... more beans to obtain service connectors
}
```

Method such as `dataSource()` come in an additional overloaded variant that offer specifying configuration options such as the pooling parameters. See Javadoc for more details.

Connect to Generic Services

Java config supports access to generic services through the `service()` method. Generic services do not have a directly mapped method. This is typical for a newly introduced service or when connecting to a private service in private PaaS. The generic `service()` method follows the same pattern as the `dataSource()`, except it allows supplying the connector type as an additional parameters.

Scan for Services

You can scan for each bound service using the `@ServiceScan` annotation as shown below. This is conceptually similar to the `@ComponentScan` annotation in Spring.
Here, one bean of the appropriate type (DataSource for a relational database service, for example) is created. Each created bean will have the id matching the corresponding service name. You can then inject such beans using auto-wiring:

```java
@Autowired DataSource inventoryDb;
```

If the app is bound to more than one services of a type, you can use the @Qualifier annotation supplying it the name of the service as in the following code:

```java
@Autowired @Qualifier("inventory-db") DataSource inventoryDb;
@Autowired @Qualifier("shipping-db") DataSource shippingDb;
```

### Access Service Properties

You can expose raw properties for all services and the app through a bean as follows:

```java
public class CloudPropertiesConfig extends AbstractCloudConfig {
  @Bean
  public Properties cloudProperties() {
    return properties();
  }
}
```

### Cloud Profile

Spring Framework versions 3.1 and above support bean definition profiles as a way to conditionalize the application configuration so that only specific bean definitions are activated when a certain condition is true. Setting up such profiles makes your application portable to many different environments so that you do not have to manually change the configuration when you deploy it to, for example, your local environment and then to Cloud Foundry.

See the Spring Framework documentation for additional information about using Spring bean definition profiles.

When you deploy a Spring application to Cloud Foundry, Cloud Foundry automatically enables the cloud profile.

> **Note:** Cloud Foundry auto-reconfiguration requires the Spring application to be version 3.1 or later and include the Spring context JAR. If you are using an earlier version, update your framework or use a custom buildpack.

### Profiles in Java Configuration

The @Profile annotation can be placed on @Configuration classes in a Spring application to set conditions under which configuration classes are invoked. By using the default and cloud profiles to determine whether the application is running on Cloud Foundry or not, your Java configuration can support both local and cloud deployments using Java configuration classes like these:
public class Configuration {
    @Configuration
    @Profile("cloud")
    static class CloudConfiguration {
        @Bean
        public DataSource dataSource() {
            CloudFactory cloudFactory = new CloudFactory();
            Cloud cloud = cloudFactory.getCloud();
            return cloud.getSingletonServiceConnector(DataSource.class, null);
        }
    }
}

@Configuration
@Profile("default")
static class LocalConfiguration {
    @Bean
    public DataSource dataSource() {
        BasicDataSource dataSource = new BasicDataSource();
        dataSource.setUrl("jdbc:postgresql://localhost/db");
        dataSource.setDriverClassName("org.postgresql.Driver");
        dataSource.setUsername("postgres");
        dataSource.setPassword("postgres");
        return dataSource;
    }
}
A Spring application can take advantage of these Cloud Foundry properties using the property placeholder mechanism. For example, assume that you have bound a MySQL service called `spring-mysql` to your application. Your application requires a `c3p0` connection pool instead of the connection pool provided by Cloud Foundry, but you want to use the same connection properties defined by Cloud Foundry for the MySQL service - in particular the username, password and JDBC URL.

The following table lists all the application properties that Cloud Foundry exposes to deployed applications.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud.application.name</td>
<td>The name provided when the application was pushed to Cloud Foundry.</td>
</tr>
<tr>
<td>cloud.provider.url</td>
<td>The URL of the cloud hosting the application, such as <code>cloudfoundry.com</code>.</td>
</tr>
</tbody>
</table>

The service properties that are exposed for each type of service are listed in the Service-Specific Details section.

### Service-Specific Details

The following sections describe Spring auto-reconfiguration and manual configuration for the services supported by Cloud Foundry.

#### MySQL and Postgres

**Auto-Reconfiguration**

Auto-reconfiguration occurs if Cloud Foundry detects a `javax.sql.DataSource` bean in the Spring application context. The following snippet of a Spring application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-reconfigure:

```xml
<bean class="org.apache.commons.dbcp.BasicDataSource"
     destroy-method="close" id="dataSource">
    <property name="driverClassName" value="org.h2.Driver"/>
    <property name="url" value="jdbc:h2:mem:"></property>
    <property name="username" value="sa"></property>
    <property name="password" value=""></property>
</bean>
```

The relational database that Cloud Foundry actually uses depends on the service instance you explicitly bind to your application when you deploy it: MySQL or Postgres. Cloud Foundry creates either a commons DBCP or Tomcat datasource depending on which datasource implementation it finds on the classpath.

Cloud Foundry internally generates values for the following properties: `driverClassName`, `url`, `username`, `password`, `validationQuery`.

**Manual Configuration in Java**

To configure a database service in Java configuration, create a `@Configuration` class with a `@Bean` method to return a `javax.sql.DataSource` bean. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```java
@Configuration
public class DataSourceConfig {
    @Bean
    public DataSource dataSource() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        return cloud.getSingletonServiceConnector(DataSource.class, null);
    }
}
```

#### MongoDB

**Auto-Reconfiguration**
You must use Spring Data MongoDB 1.0 M4 or later for auto-reconfiguration to work.

Auto-reconfiguration occurs if Cloud Foundry detects a `org.springframework.data.document.mongodb.MongoDbFactory` bean in the Spring application context. The following snippet of a Spring XML application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-reconfigure:

```xml
<mongo:db-factory
    id="mongoDbFactory"
    dbname="pwdtest"
    host="127.0.0.1"
    port="1234"
    username="test_user"
    password="test_pass"/>
```

Cloud Foundry creates a `SimpleMongoDbFactory` with its own values for the following properties: `host`, `port`, `username`, `password`, `dbname`.

Manual Configuration in Java

To configure a MongoDB service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.data.mongodb.MongoDbFactory` bean from Spring Data MongoDB. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```java
@Configuration
public class MongoConfig {

    @Bean
    public MongoDbFactory mongoDbFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        return cloud.getServiceConnector(MongoDbFactory.class, null);
    }

    @Bean
    public MongoTemplate mongoTemplate() {
        return new MongoTemplate(mongoDbFactory());
    }
}
```

Redis

Auto-Configuration

You must be using Spring Data Redis 1.0 M4 or later for auto-configuration to work.

Auto-configuration occurs if Cloud Foundry detects a `org.springframework.data.redis.connection.RedisConnectionFactory` bean in the Spring application context. The following snippet of a Spring XML application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-configure:

```xml
<bean id="redis"
    class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"
    p:hostName="localhost" p:port="6379"/>
```

Cloud Foundry creates a `JedisConnectionFactory` with its own values for the following properties: `host`, `port`, `password`. This means that you must package the Jedis JAR in your application. Cloud Foundry does not currently support the JRedis and RJC implementations.

Manual Configuration in Java

To configure a Redis service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.data.redis.connection.RedisConnectionFactory` bean from Spring Data Redis. The bean can be created by helper classes in the `spring-cloud` library, as shown here:
```java
@Configuration
public class RedisConfig {

    @Bean
    public RedisConnectionFactory redisConnectionFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        return cloud.getSingletonServiceConnector(RedisConnectionFactory.class, null);
    }

    @Bean
    public RedisTemplate redisTemplate() {
        return new StringRedisTemplate(redisConnectionFactory());
    }
}
```

RabbitMQ

Auto-Configuration

You must be using Spring AMQP 1.0 or later for auto-configuration to work. Spring AMQP provides publishing, multi-threaded consumer generation, and message conversion. It also facilitates management of AMQP resources while promoting dependency injection and declarative configuration.

Auto-configuration occurs if Cloud Foundry detects an `org.springframework.amqp.rabbit.connection.ConnectionFactory` bean in the Spring application context. The following snippet of a Spring application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-configure:

```xml
<rabbit:connection-factory
    id="rabbitConnectionFactory"
    host="localhost"
    password="testpwd"
    port="1238"
    username="testuser"
    virtual-host="/">
```

Cloud Foundry creates an `org.springframework.amqp.rabbit.connection.CachingConnectionFactory` with its own values for the following properties: host, virtual-host, port, username, password.

Manual Configuration in Java

To configure a RabbitMQ service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.amqp.rabbit.connection.ConnectionFactory` bean from the Spring AMQP library. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```java
@Configuration
public class RabbitConfig {

    @Bean
    public ConnectionFactory rabbitConnectionFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        return cloud.getSingletonServiceConnector(ConnectionFactory.class, null);
    }

    @Bean
    public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
        return new RabbitTemplate(connectionFactory);
    }
}
```
Introduction

This is a guide to using the Cloud Foundry Java Client Library to manage an account on a Cloud Foundry instance.

Note: The 1.1.x versions of the Cloud Foundry Java Client Library work with apps using Spring 4.x, and the 1.0.x versions of the Cloud Foundry Java Client Library work with apps using Spring 3.x. Both versions are available in the source repository on GitHub.

Adding the Library

Visit the Cloud Foundry Java Client Library GitHub page to obtain the correct components.

Most projects need two dependencies: the Operations API and an implementation of the Client API. Refer to the following sections for more information about how to add the Cloud Foundry Java Client Library as dependencies to a Maven or Gradle project.

Maven

Add the `cloudfoundry-client-reactor` dependency (formerly known as `cloudfoundry-client-spring`) to your `pom.xml` as follows:

```xml
<dependencies>
  <dependency>
    <groupId>org.cloudfoundry</groupId>
    <artifactId>cloudfoundry-client-reactor</artifactId>
    <version>2.0.0.BUILD-SNAPSHOT</version>
  </dependency>
  <dependency>
    <groupId>org.cloudfoundry</groupId>
    <artifactId>cloudfoundry-operations</artifactId>
    <version>2.0.0.BUILD-SNAPSHOT</version>
  </dependency>
  <dependency>
    <groupId>io.projectreactor</groupId>
    <artifactId>reactor-core</artifactId>
    <version>2.5.0.BUILD-SNAPSHOT</version>
  </dependency>
  <dependency>
    <groupId>io.projectreactor</groupId>
    <artifactId>reactor-netty</artifactId>
    <version>2.5.0.BUILD-SNAPSHOT</version>
  </dependency>
</dependencies>
```

The artifacts can be found in the Spring release and snapshot repositories:

```xml
<repositories>
  <repository>
    <id>spring-releases</id>
    <name>Spring Releases</name>
    <url>http://repo.spring.io/release</url>
  </repository>
  ... 
</repositories>
```
Gradle

Add the `cloud foundry-client-reactor` dependency to your `build.gradle` file as follows:

```groovy
dependencies {
    compile 'org.cloudfoundry:cloudfoundry-client-reactor:2.0.0.BUILD-SNAPSHOT'
    compile 'org.cloudfoundry:cloudfoundry-operations:2.0.0.BUILD-SNAPSHOT'
    compile 'io.projectreactor:reactor-core:2.5.0.BUILD-SNAPSHOT'
    compile 'io.projectreactor:reactor-netty:2.5.0.BUILD-SNAPSHOT'
    ...}
```

The artifacts can be found in the Spring release and snapshot repositories:

```groovy
repositories {
    maven { url 'http://repo.spring.io/release' }
    ...}
```

```groovy
repositories {
    maven { url 'http://repo.spring.io/snapshot' }
    ...}
```

Sample Code

The following is a very simple sample application that connects to a Cloud Foundry instance, logs in, and displays some information about the Cloud Foundry account. When running the program, provide the Cloud Foundry target API endpoint, along with a valid user name and password as command-line parameters.
import org.cloudfoundry.client.lib.CloudCredentials;
import org.cloudfoundry.client.lib.CloudFoundryClient;
import org.cloudfoundry.client.lib.domain.CloudApplication;
import org.cloudfoundry.client.lib.domain.CloudSpace;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;

public final class JavaSample {
    public static void main(String[] args) {
        String target = args[0];
        String user = args[1];
        String password = args[2];

        CloudCredentials credentials = new CloudCredentials(user, password);
        CloudFoundryClient client = new CloudFoundryClient(credentials, getTargetURL(target));
        client.login();

        System.out.printf("%nSpaces:%n");
        for (CloudSpace space : client.getSpaces()) {
            System.out.printf("%s	(%s)%n", space.getName(), space.getOrganization().getName());
        }

        System.out.printf("%nApplications:%n");
        for (CloudApplication application : client.getApplications()) {
            System.out.printf("%s%n", application.getName());
        }

        System.out.printf("%nServices%n");
        for (CloudService service : client.getServices()) {
            System.out.printf("%s	(%s)%n", service.getName(), service.getLabel());
        }
    }

    private static URL getTargetURL(String target) {
        try {
            return URI.create(target).toURL();
        } catch (MalformedURLException e) {
            throw new RuntimeException("The target URL is not valid: "+ e.getMessage());
        }
    }
}

For more details about the Cloud Foundry Java Client Library, visit the [source repository](#) in GitHub. The [domain package](#) shows the objects that you can query and inspect.
NGINX Buildpack

This topic describes how to push your NGINX app to Cloud Foundry and how to configure your NGINX app to use the NGINX buildpack.

Push an App

If your app contains an `nginx.conf` file, Cloud Foundry automatically uses the NGINX buildpack when you run `cf push` to deploy your app.

If your Cloud Foundry deployment does not have the NGINX buildpack installed or the installed version is outdated, run

```
cf push YOUR-APP -b https://github.com/cloudfoundry/nginx-buildpack.git
```

to deploy your app with the current buildpack. Replace `YOUR-APP` with the name of your app.

For example:

```
$ cf push my-app -b https://github.com/cloudfoundry/nginx-buildpack.git
```

Configure NGINX

We recommend that you use the default NGINX directory structure for your NGINX webserver. You can view this directory structure in the `nginx-buildpack` repository in GitHub.

The NGINX webserver setup includes the following:

- A root folder for all static web content
- A MIME type configuration file
- An NGINX configuration file
- A `buildpack.yml` YAML file that defines the version of NGINX to use. As an example, see `buildpack.yml` in the Cloud Foundry NGINX Buildpack repository in GitHub.

You should make any custom configuration changes based on these default files to ensure compatibility with the buildpack.

Create the nginx.conf File

Use the templating syntax when you create an `nginx.conf` file. This templating syntax loads modules and binds to ports based on values known at launch time.

Port

Use `{{port}}` to set the port to listen on. At launch time, `{{port}}` will interpolate in the value of `$PORT`.

```
server {
  listen {{port}};
}
```

Note: You must use `{{port}}` in your `nginx.conf` file.

For example, to set an NGINX server to listen on `$PORT`, include the following in your `nginx.conf` file:

```
server {
  listen {{port}};
}
```

Environment Variables
To use an environment variable, include `{{env "YOUR-VARIABLE"}}`. Replace `YOUR-VARIABLE` with the name of an environment variable. At staging and at launch, the current value of the environment variable is retrieved.

For example, include the following in your `nginx.conf` file to enable or disable GZipping based on the value of `GZIP_DOWNLOADS`:

```nginx
gzip {{env "GZIP_DOWNLOADS"}};
```

- If you set `GZIP_DOWNLOADS` to `off`, NGINX does not GZip files.
- If you set `GZIP_DOWNLOADS` to `on`, NGINX GZips files.

### Loading Dynamic Modules

To load an NGINX module, use the following syntax in your app’s `nginx.conf` file:

```nginx
{{module "MODULE-NAME"}}
```

If you have provided a module in a `modules` directory located at the root of your app, the buildpack instructs NGINX to load that module. If you have not provided a module, the buildpack instructs NGINX to search for a matching built-in module.

As of v0.0.5 of the buildpack, the `ngx_stream_module` is available as a built-in module.

For example, to load a custom module named `ngx_hello_module`, provide a `modules/ngx_hello_module.so` file in your app directory and add the following to the top of your `nginx.conf` file:

```nginx
{{module "ngx_hello_module"}}
```

To load a built-in module like `ngx_stream_module`, add the following to the top of your `nginx.conf` file. You do not need to provide an `ngx_stream_module.so` file:

```nginx
{{module "ngx_stream_module"}}
```

**Note:** To name your modules directory something other than `modules`, use the `load_module` directive, providing a path to the module relative to the location of your `nginx.conf` file. For example:

```nginx
load_module
    some_module_dir/my_module.so
```

### Buildpack Support

The following resources can assist you when using the NGINX buildpack or when developing your own NGINX buildpack:

- **NGINX Buildpack Repository in GitHub**: Find more information about using and extending the NGINX buildpack in the [NGINX buildpack](https://github.com/cloudfoundry/nginx-buildpack) GitHub repository.
- **Release Notes**: Find current information about this buildpack on the [NGINX buildpack release page](https://github.com/cloudfoundry/nginx-buildpack/releases) in GitHub.
- **Slack**: Join the #buildpacks channel in the [Cloud Foundry Slack community](https://slack.cloudfoundry.org).
.NET Core Buildpack

Page last updated:

This topic describes how to push Cloud Foundry apps to Linux cells using the .NET Core buildpack. You can find supported ASP.NET Core versions in the .NET Core buildpack release notes.

Note: The .NET Core buildpack can only be used to deploy apps to Linux cells. To deploy .NET Core apps to Windows cells, use the Binary buildpack.

Overview

Buildpacks provide needed dependencies to Pivotal Application Service for Windows (PASW) apps. PASW automatically uses the .NET Core buildpack when one or more of the following conditions are met:

- The pushed app contains one or more *.csproj or *.fsproj files.
- The app is pushed from the output directory of the dotnet publish command.

For information about deploying different types of .NET apps, follow the links in the table below.

<table>
<thead>
<tr>
<th>Type of .NET App</th>
<th>Buildpack</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP.NET MVC</td>
<td>HWC</td>
</tr>
<tr>
<td>ASP.NET Web Forms</td>
<td></td>
</tr>
<tr>
<td>ASP.NET WebAPI Apps</td>
<td></td>
</tr>
<tr>
<td>Windows Communication Foundation (WCF)</td>
<td></td>
</tr>
<tr>
<td>.NET Console</td>
<td>Binary</td>
</tr>
<tr>
<td>.NET Core pushed to Linux stack</td>
<td>.NET Core</td>
</tr>
<tr>
<td>.NET Core pushed to Windows stack</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Push a .NET Core App to Linux

Follow the steps below to push your app.

1. To push your app, run the following command:

   ```
cf push APP-NAME
   
   Where `APP-NAME` is the name you want to give your app.
   ```

   For example:

   ```
   $ cf push my-app
   Creating app my-app in org sample-org / space sample-space as username@example.com... OK
   ... requested state: started
   instances: 1/1
   usage: 1GB x 1 instances
   urls: my-app.example.com
   ```

   If your PASW deployment does not have the .NET Core buildpack installed or the installed version is out of date, run the push command with the `-b` option to specify buildpack:

   ```
cf push APP-NAME -b https://github.com/cloudfoundry/dotnet-core-buildpack.git
   
   Where `APP-NAME` is the name you want to give your app.
   ```

2. Find the URL of your app in the push command output. In the example above, `my-app.example.com` is the URL of the app.
3. Open a browser and navigate to the URL to see your app running.

For a basic example app, see [ASP.NET Core getting started app](https://github.com/dotnet) in GitHub.

## Source-Based, Non-Published Deployments

For a source-based, non-published deployment, you push your app’s source code, not the output directory of the `dotnet publish` command.

The source-based, non-published workflow ensures the buildpack can keep all your dependencies in sync and up to date. For additional information about using source-based, non-published deployments, see the following sections:

- Deploy Apps with Multiple Projects
- Use Non-Default Package Sources
- Disable and Clear Your NuGet Package Cache

> **Note:** The source-based deployment workflow also uses the `cf push` command to push source-based apps to Cloud Foundry.

## Deploy Apps with Multiple Projects

If you are deploying an app containing multiple projects, you must specify which of the app’s projects is the main project.

To specify the main project in a multi-project deployment perform the following steps:

1. Create a `.deployment` file in your app’s root folder and open the new file in a text editor.
2. Designate the main project’s path by configuring the file, using the following format:

   ```
   [config]
   project = PATH-TO-YOUR-MAIN-PROJECT
   
   Where `PATH-TO-YOUR-MAIN-PROJECT` is the location of your main project’s `*.csproj` or `*.fsproj` file.
   
   For example:
   ```
   ```
   ```
   ```
   [config]
   project = src/MyApp.Web/MyApp.Web.csproj
   ```
   ```
   ```
   In this example, by pointing to the `MyApp.Web` file, `MyApp.Web` is configured as the main project.
2. Save the revised `.deployment` file.

When deployed, the buildpack attempts to execute the main project using the `dotnet run` command, and automatically compiles all projects listed as dependencies in the main project’s `*.csproj` or `*.fsproj` file.

For example: Suppose your `.deployment` file is configured as above and you have an app in the `src` folder containing three projects: `MyApp.Web`, `MyApp.DAL` and `MyApp.Services`. If your `MyApp.Web.csproj` file lists the `MyApp.DAL` and `MyApp.Services` projects as dependencies, the two additional projects will be compiled by the buildpack.

## Use Non-Default Package Sources

If you want to deploy an app that uses non-default package sources, you must specify those package sources in the `NuGet.Config` file. For information about `NuGet.Config`, see [nuget.config reference](https://docs.microsoft.com) in the Microsoft documentation.

## Disable and Clear Your NuGet Package Cache

You may need to disable NuGet package caching or clear NuGet packages cached in the staging environment in one of the following scenarios:

© Copyright Pivotal Software Inc, 2013-2019
Your app fails to stage because it runs out of space, exceeding the maximum allowable disk quota.
You have added pre-release packages to test a new feature and then decided to revert back to the main NuGet feed. You may need to remove the packages you changed from the cache to avoid conflicts.

Disabling NuGet caching clears any existing NuGet dependencies from the staging cache and prevents the buildpack from adding NuGet dependencies to the staging cache.

NuGet package caching is disabled by default. If the default is not explicitly overridden, no additional NuGet caching configuration is required.

To disable NuGet package caching, do the following:

1. Confirm the `CACHE_NUGET_PACKAGES` environment variable is not set to `true` in your app manifest by locating the `manifest.yml` file and confirming `CACHE_NUGET_PACKAGES` is not set to `true`.
2. If needed, set `CACHE_NUGET_PACKAGES` to `false` in the `manifest.yml` file by setting the `CACHE_NUGET_PACKAGES` environment variable to `false`.

For example:

```yaml
---
applications:
  - name: sample-aspnetcore-app
    memory: 512M
    env:
      CACHE_NUGET_PACKAGES: false
---
```

3. To alternatively configure the setting to false in the environment variables settings, run the following on the command line:

```
cf set-env APP-NAME CACHE_NUGET_PACKAGES false
```

Where `APP-NAME` is the name of your app.

For more information, see the Environment Variables section of the Deploying with Application Manifests topic.

Framework-Dependent Deployments

For a framework-dependent deployment (FDD), you deploy only your app and third-party dependencies. Cloud Foundry recommends using this workflow if you deploy an app in an offline setting. For information about deploying FDDs, see Framework-dependent deployments (FDD) in the Microsoft documentation.

To deploy an FDD using the buildpack, do the following:

1. Publish the app by running the `dotnet publish` command:

```
dotnet publish [-f FRAMEWORK-NAME] [-c Release]
```

Where `FRAMEWORK-NAME` is your target framework.

2. Prepare to push your app by doing one of the following:
   - If your app uses a `manifest.yml`, specify a path to the output folder of `dotnet publish`. This allows you to push your app from any directory.
   - If not, navigate to the `bin/Debug|Release/YOUR-FRAMEWORK/YOUR-RUNTIME/publish` directory.

3. Push your app.

Self-Contained Deployments

For a self-contained deployment (SCD), you deploy your app, third-party dependencies, and the version of .NET Core that you used to build your app. For information about SCDs, see Self-contained deployments (SCD) in the Microsoft documentation.

Note: Cloud Foundry does not recommend using the SCD workflow. The buildpack is unable to keep dependencies in sync and up to date for workflows that deploy a pre-published binary.
When using the SCD workflow for deploying your app, you must do the following:

- Specify a runtime in the `dotnet publish` command. For example:

  ```bash
  $ dotnet publish -r ubuntu14.04-x64
  ```

- Include the specified runtime in the `RuntimeIdentifiers` section of the project file.

### Specify .NET Core SDKs

To pin the .NET Core SDK to a specific version or version line, create a `buildpack.yml` file at the app root and add your SDK version in one of the following formats:

- `dotnet-core:
  sdk: 2.1.201`
- `dotnet-core:
  sdk: 2.1.x`
- `dotnet-core:
  sdk: 2.x`

The buildpack chooses what SDK to install based on the files present at the app root in the following order of precedence:

1. `buildpack.yml`
2. `global.json`
3. `*.fsproj`

**Note:** The app respects the SDK version specified in `global.json` at runtime. If you provide versions in both `global.json` and `buildpack.yml` files, ensure you specify the same versions in both files.

### Specify .NET Runtime Versions

This section explains how to specify a .NET Runtime version for source-based and framework-dependent apps.

#### Source-Based Apps

If you want to lock the .NET Runtime version, configure your `*.csproj` or `*.fsproj` file to lock your app to the desired version.

For example, the following configuration locks the runtime to 2.1:

```xml
<Project>
  <PropertyGroup>
    <TargetFramework>netcoreapp2.1</TargetFramework>
    <RuntimeFrameworkVersion>2.1.*</RuntimeFrameworkVersion>
  </PropertyGroup>
</Project>
```

**Note:** For source-based apps, specify a minor version of the .NET Runtime. Do not specify a patch version because buildpacks contain only the two most recent patch versions of each minor version.

#### Framework-Dependent Apps

Your app is configured to use the latest .NET Runtime patch version by default.

If you want your app to maintain a specific .NET Runtime version, you must modify your app’s `runtimeconfig.json` file to include the `appPatches` property
and set the property to `false`.

For example:

```json
{
  "runtimeOptions": {
    "tfm": "netcoreapp2.0",
    "Framework": {
      "name": "Microsoft.NETCore.App",
      "version": "2.0.0"
    },
    "applyPatches": false
  }
}
```

Note: Set `applyPatches: false` in `*.runtimeconfig.json` only if you want to pin your .NET Framework to a specific version. This prevents your app from receiving updates to the runtime version and assemblies.

---

### Push an App in a Disconnected Environment

For offline environments, Cloud Foundry recommends using the Framework-Dependent Deployment workflow. This workflow enables the deployed app to use the latest runtime provided by the offline buildpack. For more information, see: [Framework-Dependent Deployments](#).

---

### Maintain ASP.NET Core Assemblies

Note: This section applies only to source-based and framework-dependent deployments.

For maintaining ASP.NET Core assemblies, it is recommended you do one of the following:

- Configure your app as a fully vendored app requiring fewer buildpack updates. Modify your `.csproj` file with the following:

  ```xml
  <PropertyGroup>
  <PublishWithAspNetCoreTargetManifest>false</PublishWithAspNetCoreTargetManifest>
  </PropertyGroup>
  ```

- Keep your SDK up to date, by setting `buildpack.yml` to the .NET SDK line you want to use. For example:

  ```yaml
  ---
  dotnet-core:
  sdk: 2.0.x
  ```

  Note: 2.0.x ASP.NET Core assemblies are released in the 2.1.200 - 2.1.299 SDK versions, and 2.1.x assemblies are released in the 2.1.300 and above SDK versions.

---

### Configure the Listen Port

PASW sets the `SPORT` environment variable automatically. For your .NET Core app to work on PASW, you must configure the app to listen on the environment’s specified port.

For C# apps, the following modifications allow the buildpack to pass the correct port from `SPORT` environment variable to the app when running the initial startup command:

1. Open the file that contains your `Main` method.
2. Add a `using` statement to the top of the file:

   ```csharp
   using Microsoft.Extensions.Configuration;
   ```
3. Add the following lines before the line `var host = new WebHostBuilder()`:

   ```csharp
   var port = Environment.GetEnvironmentVariable("SPORT");
   ```

© Copyright Pivotal Software Inc, 2013-2019 1407 2.5
4. Add the following line after `UseKestrel();`:

```
.UseConfiguration(config)
```

The `Main` method should now resemble the following example:

```
public static void Main(string[] args)
{
    var config = new ConfigurationBuilder()
        .AddCommandLine(args)
        .Build();
    var host = new WebHostBuilder()
        .UseKestrel()
        .UseConfiguration(config)
        .UseContentRoot(Directory.GetCurrentDirectory())
        .UseStartup<Startup>();
    host.Run();
}
```

5. Save your changes.


```
    <Version>VERSION</Version>
</PackageReference>
```

Where `VERSION` is the version of the package to use. To find a list of valid versions, navigate to [https://www.nuget.org](https://www.nuget.org).

7. If your app requires any other files at runtime, such as JSON configuration files, add them to the `include` section of `copyToOutput`.

8. Save your changes.

With these changes, the `dotnet run` command copies your app `Views` to the build output where the .NET CLI can find them.

### Add Custom Libraries

If your app requires external shared libraries that are not provided by the rootfs or the buildpack, you must place the libraries in an `ld_library_path` directory at the app root.

**Note:** You must keep these libraries up to date. They do not update automatically.

The .NET Core buildpack automatically adds the directory `/<app-root>/ld_library_path` to `LD_LIBRARY_PATH` so that your app can access these libraries at runtime.
Node.js Buildpack

Page last updated:

Use the Node.js buildpack with Node or JavaScript apps.

You must install the Cloud Foundry Command Line Interface tool (cf CLI) to run some of the commands listed in this topic.

Push Node.js Apps

Cloud Foundry automatically uses the Node.js buildpack if it detects a `package.json` file in the root directory of your project.

The `--buildpack` option lets you specify a buildpack to use with the `cf push` command. If your Cloud Foundry deployment does not have the Node.js buildpack installed or the installed version is out of date, run `cf push APP-NAME --buildpack https://github.com/cloudfoundry/nodejs-buildpack`, where `APP-NAME` is the name of your app, to push your app with the latest version of the buildpack.

For example:

```bash
$ cf push my-nodejs-app --buildpack https://github.com/cloudfoundry/nodejs-buildpack
```

For more detailed information about deploying Node.js apps, see the following topics:

- Tips for Node.js Developers
- Environment Variables Defined by the Node Buildpack
- Configuring Service Connections for Node
- Node.js Buildpack Source Code on GitHub

Supported Versions

For a list of supported Node versions, see the Node.js Buildpack release notes on GitHub.

Specify a Node.js Version

To specify a Node.js version, set the `engines.node` in the `package.json` file to the semantic versioning specification (semver) range or the specific version of Node you are using.

Example showing a semver range:

```json
"engines": {  
  "node": "6.9.x"
}
```

Example showing a specific version:

```json
"engines": {  
  "node": "6.9.0"
}
```

If you try to use a version that is not currently supported, staging your app fails with the following error message:

```bash
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:
  |
  | exit
  | Staging failed: Buildpack compilation step failed
```
Specify an npm Version

To specify an npm version, set `engines.npm` in the `package.json` file to the semantic versioning specification (semver) range or the specific version of npm you are using:

Example showing a semver range:

```
"engines": {
  "node": "6.9.x",
  "npm": "2.15.x"
}
```

Example showing a specific version:

```
"engines": {
  "node": "6.9.0",
  "npm": "2.15.1"
}
```

If you do not specify an npm version, your app uses the default npm packaged with the Node.js version used by your app, as specified on the [Node.js releases](#) page.

If your environment cannot connect to the Internet and you specified a non-supported version of npm, the buildpack fails to download npm and you see the following error message:

```
We're unable to download the version of npm you've provided (...).
Please remove the npm version specification in package.json (...)
Staging failed: Buildpack compilation step failed
```

Using NPM or Yarn

By default, the Node.js buildpack assumes you are using NPM. If you want to use Yarn instead, you must provide a `yarn.lock` in your top-level app directory. For more information, see [yarn.lock](#) in the Yarn documentation.

Vendororing App Dependencies

To vendor dependencies for an app using the Node.js buildpack, run `npm install` (or `yarn install`, if you are using Yarn) from your app directory. This command vendors dependencies into the `node_modules` directory of your app directory.

For example, the following example vendors dependencies into the `my-nodejs-app/node_modules` directory:

```
$ cd my-nodejs-app
$ npm install
```

The `cf push` command uploads the vendored dependencies with the app.

**Note:** For an app to run in a disconnected environment, it must vendor its dependencies and provide a lock file.

Usage in Offline Environments

When vendororing apps for usage in offline environments, you must supply a lock file. For information about npm lockfiles, see [npm-package-locks](#) in the NPM documentation. For information about the NPM lock file, see [package-lock.json](#) in the NPM documentation. For information about the Yarn lock file, see [yarn.lock](#) in the Yarn documentation.

This lock file informs the package manager of the exact versions of dependencies and transitive dependencies to look for when running `npm install` or `yarn install`. For this reason, you must keep `package.json` the vendored `node_modules` directory, and your lock file synchronized to avoid network calls.

**Note:** `package-lock.json` is only supported by NPM 5.x and later. For earlier versions of NPM, provide a `npm-shrinkwrap.json` file instead.
Yarn Offline Mode

Versions 1.5.28 and later of the Node.js buildpack include the ability to use Yarn in offline mode. To do so, you must mirror the Yarn registry locally by providing an `npm-packages-offline-cache` directory:

```
$ cd APP-DIR
$ yarn config set yarn-offline-mirror ./npm-packages-offline-cache
$ yarn config set yarn-offline-mirror-pruning true
$ rm -rf node_modules/yarn.lock # if they were previously generated
$ yarn install
```

When you push the app, the buildpack looks for an `npm-packages-offline-cache` directory at the top level of the app directory. If this directory exists, the buildpack runs Yarn in offline mode. Otherwise, it runs Yarn normally, which may require an Internet connection. You do not have to provide a `node_modules` directory when running Yarn in offline mode, as the offline cache will provide the dependencies.

For more information about using an offline mirror with Yarn, see the Yarn Blog.

Integrity Check

By default, the Node.js buildpack uses npm to download dependencies. Note that npm does not perform integrity checks of the downloaded packages, which is a security risk.

If missing dependencies are detected, the buildpack runs `npm install` for non-vendored dependencies or `npm rebuild` for dependencies that are already vendored. This may result in code being downloaded and executed without verification.

You can use Yarn as an alternative that verifies dependencies.

OpenSSL Support

The `nodejs-buildpack` packages binaries of Node.js with OpenSSL that are statically linked. The Node.js buildpack supports Node.js 4.x and later, which relies on the Node.js release cycle to provide OpenSSL updates. The `binary-builder` enables static OpenSSL compilation.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH Configured Custom Trusted Certificate Support

Node.js hardcodes root CA certs in its source code. To use BOSH configured custom trusted certificates, a developer must pass the specified CAs to the `tls.connect` function as extra arguments.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Node.js buildpack in Cloud Foundry, see the Node.js GitHub repository.

You can find current information about this buildpack in the Node.js buildpack release page in GitHub.
Tips for Node.js Applications

This topic provides Node-specific information to supplement the general guidelines in the Deploying an Application topic.

About the Node.js Buildpack

For information about using and extending the Node.js buildpack in Cloud Foundry, see the nodejs-buildpack repository in GitHub.

You can find current information about this buildpack on the Node.js buildpack release page in GitHub.

The buildpack uses a default Node.js version. To specify the versions of Node.js and npm an app requires, edit the app’s package.json, as described in “node.js and npm versions” in the nodejs-buildpack repository.

Application Package File

Cloud Foundry expects a package.json in your Node.js app. You can specify the version of Node.js you want to use in the engine node of your package.json file.

In general, Cloud Foundry supports the two most recent versions of Node.js. See the GitHub Node.js buildpack page for current information.

Example package.json file:

```json
{
    "name": "first",
    "version": "0.0.1",
    "author": "Demo",
    "dependencies": {
        "express": "3.4.8",
        "consolidate": "0.10.0",
        "swig": "1.3.2"
    },
    "engines": {
        "node": "0.12.7",
        "npm": "2.7.4"
    }
}
```

Application Port

You must use the PORT environment variable to determine which port your app should listen on. To also run your app locally, set the default port as 3000.

```javascript
app.listen(process.env.PORT || 3000);
```

Low Memory Environments

When running node apps, you might notice that instances are occasionally restarted due to memory constraints. Node does not know how much memory it is allowed to use, and thus sometimes allows the garbage collector to wait past the allowed amount of memory. To resolve this issue, set the OPTIMIZE_MEMORY environment variable to true (requires node v6.12.0 or greater). This sets max_old_space_size based on the available memory in the instance.

```bash
$ cf set-env my-app OPTIMIZE_MEMORY true
```
Application Start Command

Node.js apps require a start command. You can specify the web start command for a Node.js app in a Procfile or in the app deployment manifest. For more information about Procfiles, see the Configuring a Production Server topic.

The first time you deploy, you are asked if you want to save your configuration. This saves a manifest.yml in your app with the settings you entered during the initial push. Edit the manifest.yml file and create a start command as follows:

```yaml
---
applications:
  - name: my-app
    command: node my-app.js
    ...
    the rest of your settings ...
```

Alternately, specify the start command with cf push -c "node my-app.js"

```bash
$ cf push my-app -c "node my-app.js"
```

Application Bundling

You do not need to run npm install before deploying your app. Cloud Foundry runs it for you when your app is pushed. You can, if you prefer, run npm install and create a node_modules folder inside of your app.

Solve Discovery Problems

If Cloud Foundry does not automatically detect that your app is a Node.js app, you can override auto-detection by specifying the Node.js buildpack.

Add the buildpack into your manifest.yml and re-run cf push with your manifest:

```yaml
---
applications:
  - name: my-app
    buildpack: https://github.com/cloudfoundry/nodejs-buildpack
    ...
    the rest of your settings ...
```

Alternately, specify the buildpack on the command line with cf push -b:

```bash
$ cf push my-app -b https://github.com/cloudfoundry/nodejs-buildpack
```

Bind Services

Refer to Configure Service Connections for Node.js.

Environment Variables

You can access environments variable programmatically.

For example, you can obtain VCAP_SERVICES as follows:

```javascript
process.env.VCAP_SERVICES
```

Environment variables available to you include both those defined by the system and those defined by the Node.js buildpack, as described below.
BUILD_DIR
Directory into which Node.js is copied each time a Node.js app is run.

CACHE_DIR
Directory that Node.js uses for caching.

PATH
The system path used by Node.js.

PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin
Environment Variables Defined by the Node Buildpack

Page last updated:

Pivotal Application Service (PAS) provides configuration information to applications through environment variables. This topic describes the additional environment variables provided by the Node buildpack.

For more information about the standard environment variables provided by Pivotal Application Service (PAS), see the Cloud Foundry Environment Variables topic.

Node Buildpack Environment Variables

The following table describes the environment variables provided by the Node buildpack.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD_DIR</td>
<td>The directory where Node.js is copied each time a Node.js application runs.</td>
</tr>
<tr>
<td>CACHE_DIR</td>
<td>The directory Node.js uses for caching.</td>
</tr>
<tr>
<td>PATH</td>
<td>The system path used by Node.js: <code>PATH=/home/vcap/bin:/home/vcap/node_modules/.bin:/bin:/usr/bin</code></td>
</tr>
</tbody>
</table>
Configuring Service Connections for Node.js

This guide is for developers who wish to bind a data source to a Node.js application deployed and running on Cloud Foundry.

Parse VCAP_SERVICES for Credentials

You must parse the VCAP_SERVICES environment variable in your code to get the required connection details such as host address, port, user name, and password.

For example, if you are using PostgreSQL, your VCAP_SERVICES environment variable might look something like this:

```json
{
  "mypostgres": [
    {
      "name": "myinstance",
      "credentials": {
        "uri": "postgres://myusername:mypassword@host.example.com:5432/serviceinstance"
      }
    }
  ]
}
```

This example JSON is simplified; yours may contain additional properties.

Parse with cfenv

The cfenv package provides access to Cloud Foundry application environment settings by parsing all the relevant environment. The settings are returned as JavaScript objects. cfenv provides reasonable defaults when running locally, as well as when running as a Cloud Foundry application.

- [https://www.npmjs.org/package/cfenv](https://www.npmjs.org/package/cfenv)

Manual Parsing

First, parse the VCAP_SERVICES environment variable.

For example:

```javascript
var vcap_services = JSON.parse(process.env.VCAP_SERVICES)
```

Then pull out the credential information required to connect to your service. Each service package requires different information. If you are working with Postgres, for example, you will need a `uri` to connect. You can assign the value of the `uri` to a variable as follows:

```javascript
var uri = vcap_services.mypostgres[0].credentials.uri
```

Once assigned, you can use your credentials as you would normally in your program to connect to your database.

Connecting to a Service

You must include the appropriate package for the type of services your application uses. For example:

- Rabbit MQ via the amqp module
- Mongo via the mongodb and mongoose modules
- MySQL via the mysql module
- Postgres via the pg module
- Redis via the redis module
Add the Dependency to package.json

Edit `package.json` and add the intended module to the `dependencies` section. Normally, only one would be necessary, but for the sake of the example we will add all of them:

```json
{
  "name": "hello-node",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "mongodb": "*",
    "mongoose": "*",
    "mysql": "*",
    "pg": "*",
    "redis": "*",
    "amqp": "*"
  },
  "engines": {
    "node": "0.8.x"
  }
}
```

You must run `npm shrinkwrap` to regenerate your `npm-shrinkwrap.json` file after you edit `package.json`. 

PHP Buildpack

Page last updated:

Use the PHP buildpack with PHP or HHVM runtimes.

Supported Software and Versions

The release notes page has a list of currently supported modules and packages.

- **PHP Runtimes**
  - php-cli
  - php-cgi
  - php-fpm

- **Third-Party Modules**
  - New Relic, in connected environments only.

Push an App

30 Second Tutorial

Getting started with this buildpack is easy. With the Cloud Foundry Command Line Interface installed, open a shell, change directories to the root of your PHP files and push your application using the argument `-b https://github.com/cloudfoundry/php-buildpack.git`.

Example:

```bash
$ mkdir my-php-app
$ cd my-php-app
$ cat <<EOF > index.php
<?php
phpinfo();
?>
EOF
$ cf push -m 128M -b https://github.com/cloudfoundry/php-buildpack.git my-php-app
```

Change `my-php-app` in the above example to a unique name on your target Cloud Foundry instance to prevent a hostname conflict error and failed push.

The example above creates and pushes a test application, `my-php-app`, to Cloud Foundry. The `-b` argument instructs CF to use this buildpack. The remainder of the options and arguments are not specific to the buildpack, for questions on those consult the output of `cf help push`.

Here’s a breakdown of what happens when you run the example above.

- **On your PC:**
  - It will create a new directory and one PHP file, which just invokes `phpinfo()`.
  - Run `cf` to push your application. This will create a new application with a memory limit of 128M (more than enough here) and upload our test file.

- **Within Cloud Foundry:**
  - The buildpack is executed.
  - Application files are copied to the `htdocs` folder.
  - Apache HTTPD & PHP are downloaded, configured with the buildpack defaults and run.
  - Your application is accessible at the URL [http://my-php-app.example.com](http://my-php-app.example.com) (Replacing `example.com` with the domain of your public CF provider or private instance).

More information about deploying
While the 30 Second Tutorial shows how quick and easy it is to get started using the buildpack, it skips over quite a bit of what you can do to adjust, configure and extend the buildpack. The following sections and links provide a more in-depth look at the buildpack.

Features

Here are some special features of the buildpack.

- Supports running commands or migration scripts prior to application startup.
- Supports an extension mechanism that allows the buildpack to provide additional functionality.
- Allows for application developers to provide custom extensions.
- Easy troubleshooting with the `BP_DEBUG` environment variable.
- Download location is configurable, allowing users to host binaries on the same network (i.e. run without an Internet connection)
- Smart session storage, defaults to file w/sticky sessions but can also use redis for storage.

Examples

Here are some example applications that can be used with this buildpack.

- **php-info** This app has a basic index page and shows the output of `phpinfo()`.
- **PHPMyAdmin** A deployment of PHPMyAdmin that uses bound MySQL services.
- **PHPPgAdmin** A deployment of PHPPgAdmin that uses bound PostgreSQL services.
- **Drupal** A deployment of Drupal that uses bound MySQL service.
- **CodeIgniter** CodeIgniter tutorial application running on CF.
- **Stand Alone** An example which runs a standalone PHP script.
- **pgbouncer** An example which runs the PgBouncer process in the container to pool database connections.
- **phalcon** An example which runs a Phalcon based application.
- **composer** An example which uses Composer.

Advanced Topics

See the following topics:

- **Tips for PHP Developers**
- **Getting Started Deploying PHP Apps**
- **PHP Buildpack Configuration**
- **Composer**
- **Sessions**
- **New Relic**

You can find the source for the buildpack on GitHub: [https://github.com/cloudfoundry/php-buildpack](https://github.com/cloudfoundry/php-buildpack)

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see [Using a Proxy](#).

BOSH Configured Custom Trusted Certificate Support

For versions of PHP 5.6.0 and later, the default certificate location is `/usr/lib/ssl/certs`, which symlinks to `/etc/ssl/certs`. Your platform operator can configure the platform to [add the custom certificates into the application container](#).
Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the PHP buildpack in Cloud Foundry, see the php-buildpack GitHub repository.

You can find current information about this buildpack on the PHP buildpack release page in GitHub.

License

The Cloud Foundry PHP Buildpack is released under version 2.0 of the Apache License.
Tips for PHP Developers

Page last updated:

About the PHP Buildpack

For information about using and extending the PHP buildpack in Cloud Foundry, see the [php-buildpack GitHub repository](https://github.com/cloudfoundry/php-buildpack).

You can find current information about this buildpack on the [PHP buildpack release page](https://github.com/cloudfoundry/php-buildpack/releases) in GitHub.

The buildpack uses a default PHP version specified in [defaults/options.json](https://github.com/cloudfoundry/php-buildpack#release) under the `PHP_VERSION` key.

To change the default version, specify the `PHP_VERSION` key in your app's [options.json](https://github.com/cloudfoundry/php-buildpack#release) file.
Getting Started Deploying PHP Apps

Prerequisites

- Basic PHP knowledge
- The Cloud Foundry Command Line Interface (cf CLI) installed on your workstation

A First PHP Application

```bash
mkdir my-php-app
cd my-php-app
cat <<EOF >index.php
<?php
phpinfo();
?>
EOF
cf push my-php-app -m 128M
```

Change “my-php-app” to a unique name or you may see an error and a failed push.

The example above creates and pushes a test application to Cloud Foundry.

Here is a breakdown of what happens when you run the example above:

- On your workstation…
  - It creates a new directory and one PHP file, which calls `phpinfo()`
  - Run `cf push` to push your application. This will create a new application with a memory limit of 128M and upload our test file.

- On Cloud Foundry…
  - The buildpack detects that your app is a php app
  - The buildpack is executed.
    - Application files are copied to the `htdocs` folder.
  - Apache HTTPD & PHP are downloaded, configured with the buildpack defaults, and run.
  - Your application is accessible at the default route. Use `cf app my-php-app` to view the url of your new app.

Folder Structure

The easiest way to use the buildpack is to put your assets and PHP files into a directory and push it to PAS. This way, the buildpack will take your files and automatically move them into the `WEBDIR` (defaults to `htdocs`) folder, which is the directory where your chosen web server looks for the files.

URL Rewriting

If you select Apache as your web server, you can include `.htaccess` files with your application.

Alternatively, you can provide your own Apache or Nginx configurations.
Prevent Access To PHP Files

The buildpack will put all of your files into a publicly accessible directory. In some cases, you might want to have PHP files that are not publicly accessible but are on the `include_path` path. To do that, create a `/lib` directory in your project folder and place your protected files there.

For example:

```
$ ls -lRh
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:40 images
-rw-r--r--  1 daniel  staff  08 Feb 27 21:39 index.php
-dwxr-xr-x  3 daniel  staff  102B Feb 27 21:40 lib

./lib:
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:40 my.class.php  <-- not public, http://app.cfapps.io/lib/my.class.php == 404
```

This comes with a catch. If your project legitimately has a `lib` directory, these files will not be publicly available because the buildpack does not copy a top-level `lib` directory into the `WEBDIR` folder. If your project has a `lib` directory that needs to be publicly available, then you have two options as follows:

Option #1

In your project folder, create an `.htdocs` folder (or whatever you’ve set for `WEBDIR`). Then move any files that should be publicly accessible into this directory. In the example below, the `lib/controller.php` file is publicly accessible.

Example:

```
$ ls -lRh
total 0
dwrxr-xr-x  7 daniel  staff  238B Feb 27 21:48 .htdocs

./.htdocs:  <-- create the .htdocs directory and put your files there
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:40 images
-rw-r--r--  1 daniel  staff  08 Feb 27 21:39 index.php
-dwxr-xr-x  3 daniel  staff  102B Feb 27 21:48 lib

./.htdocs/lib:  <-- anything under .htdocs is public, including a lib directory
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:48 controller.php
```

Given this setup, it is possible to have both a public `lib` directory and a protected `lib` directory. The following example demonstrates this setup:

Example:

```
$ ls -lRh
total 0
dwrxr-xr-x  7 daniel  staff  238B Feb 27 21:48 .htdocs
dwrxr-xr-x  3 daniel  staff  102B Feb 27 21:51 lib

./.htdocs:
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:40 images
-rw-r--r--  1 daniel  staff  08 Feb 27 21:39 index.php
-dwxr-xr-x  3 daniel  staff  102B Feb 27 21:48 lib

./.htdocs/lib:  <-- public lib directory
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:48 controller.php

lib:  <-- protected lib directory
total 0
-rw-r--r--  1 daniel  staff  08 Feb 27 21:51 my.class.php
```

Option #2

The second option is to pick a different name for the `LIBDIR`. This is a configuration option that you can set (it defaults to `lib`). Thus if you set it to something else such as `include`, your application’s `lib` directory would no longer be treated as a special directory and it would be placed into `WEBDIR`.
(i.e. become public).

Other Folders

Beyond the \WEBDIR\ and \LHDDIR\ directories, the buildpack also supports a \bp-config\ directory and a \extensions\ directory.

The \bp-config\ directory should exist at the root of your project directory and it is the location of application-specific configuration files. Application-specific configuration files override the default settings used by the buildpack. This link explains application configuration files in depth.

The \extensions\ directory should also exist at the root of your project directory and it is the location of application-specific custom extensions. Application-specific custom extensions allow you, the developer, to override or enhance the behavior of the buildpack. See the php-buildpack README in GitHub for more information about extensions.

Troubleshooting

To troubleshoot problems using the buildpack, you can do one of the following:

1. Review the output from the buildpack. The buildpack writes basic information to stdout, for example the files that it downloads. The buildpack also writes information in the form of stack traces when a process fails.

2. Review the logs from the buildpack. The buildpack writes logs to disk. Follow the steps below to access these logs:

   a. Run `cf ssh APP-NAME` to ssh into the app container, replacing `APP-NAME` with the name of your app.
   b. Run `cat app/.bp/logs/bp.log` to view the logs.

By default, log files contain more detail than output to stdout. Set the `BP_DEBUG` environment variable to `true` for more verbose logging. This instructs the buildpack to set its log level to `DEBUG`, and to output logs to stdout. Follow Environment Variables documentation to set `BP_DEBUG`.

Increase Log Output with fpm.d

If you use fpm.d, follow the steps below to configure fpm to redirect worker stdout and stderr into the logs.

1. Create a file in the \hp-config\php\fpm.d\ directory of your app.

2. Add `catch_workers_output=yes` to the file you created.

3. Push your app.

For more information about allowed configuration settings in the \hp-config\php\fpm.d\ directory, see the List of global php-fpm.conf directives.

You can see an example fmp fixture and configuration file in the php-buildpack GitHub repository.
PHP Buildpack Configuration

Page last updated:

Defaults

The PHP buildpack stores all of its default configuration settings in the defaults directory.

options.json

The options.json file is the configuration file for the buildpack itself. It instructs the buildpack what to download, where to download it from, and how to install it. It allows you to configure package names and versions (i.e. PHP, HTTPD, or Nginx versions), the web server to use (HTTPD, Nginx, or None), and the PHP extensions that are enabled.

The buildpack overrides the default options.json file with any configuration it finds in the .bp-config/options.json file of your application.

Below is an explanation of the common options you might need to change.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB_SERVER</td>
<td>Sets the web server to use. Must be one of: httpd, nginx, or none. This value defaults to httpd.</td>
</tr>
<tr>
<td>HTTPD_VERSION</td>
<td>Sets the version of Apache HTTPD to use. Currently the build pack supports the latest stable version. This value will default to the latest release that is supported by the build pack.</td>
</tr>
<tr>
<td>ADMIN_EMAIL</td>
<td>The value used in HTTPD's configuration for ServerAdmin.</td>
</tr>
<tr>
<td>NGINX_VERSION</td>
<td>Sets the version of Nginx to use. By default, the buildpack uses the latest stable version.</td>
</tr>
<tr>
<td>PHP_VERSION</td>
<td>Sets the version of PHP to use. Set to a minor instead of a patch version, such as &quot;(PHP_70_LATEST)&quot;. See options.json.</td>
</tr>
<tr>
<td>PHP_EXTENSIONS</td>
<td>(DEPRECATED) A list of the extensions to enable. bz2, zlib, curl, and mcrypt are enabled by default.</td>
</tr>
<tr>
<td>PHP_MODULES</td>
<td>A list of the modules to enable. No modules are explicitly enabled by default, but the buildpack automatically chooses either fpm or cli. You can explicitly enable any or all of the following: fpm, cli, cgi, and pear.</td>
</tr>
<tr>
<td>ZEND_EXTENSIONS</td>
<td>A list of the Zend extensions to enable. Nothing is enabled by default.</td>
</tr>
<tr>
<td>APP_START_CMD</td>
<td>When the WEB_SERVER option is set to none, this command is used to start your app. If WEB_SERVER and APP_START_CMD are not set, then the buildpack searches, in order, for app.php, main.php, run.php, or start.php. This option accepts arguments.</td>
</tr>
<tr>
<td>WEBDIR</td>
<td>The root directory of the files served by the web server specified in WEB_SERVER. Defaults to htdocs. Other common settings are public, static, or html. The path is relative to the root of your application.</td>
</tr>
<tr>
<td>LIBDIR</td>
<td>This path is added to PHP's include_path. Defaults to lib. The path is relative to the root of your application.</td>
</tr>
<tr>
<td>HTTP_PROXY</td>
<td>The buildpack downloads uncached dependencies using HTTP. If you are using a proxy for HTTP access, set its URL here.</td>
</tr>
<tr>
<td>HTTPS_PROXY</td>
<td>The buildpack downloads uncached dependencies using HTTPS. If you are using a proxy for HTTPS access, set its URL here.</td>
</tr>
<tr>
<td>ADDITIONAL_PREPROCESS_CMDS</td>
<td>A list of additional commands that run prior to the application starting. For example, you might use this command to run migration scripts or static caching tools before the application launches.</td>
</tr>
</tbody>
</table>

For details about supported versions, see the release notes for your buildpack version.

HTTPD, Nginx, and PHP configuration

The buildpack automatically configures HTTPD, Nginx, and PHP for your application. This section explains how to modify the configuration.

The .bp-config directory in your application can contain configuration overrides for these components. Name the directories httpd, nginx, and php. We recommend that you use php.ini.d or fpm.d.
For example:

```
$.bp-config httpd
```

Each directory can contain configuration files that the component understands.

For example, to change HTTPD logging configuration:

```
$ ls -l .bp-config/extra/
 total 8
-rw-r--r-- 1 daniel staff 396 Jan 3 08:31 httpd-logging.conf
```

In this example, the `httpd-logging.conf` file overrides the one provided by the buildpack. We recommend that you copy the default from the buildpack and modify it.

You can find the default configuration files in the `PHP Buildpack /defaults/config` directory.

You should be careful when modifying configurations, as doing so can cause your application to fail, or cause Cloud Foundry to fail to stage your application.

You can add your own configuration files. The components will not know about these, so you must ensure that they are included. For example, you can add an include directive to the `httpd configuration` to include your file:

```
ServerRoot "${HOME}/httpd"
Listen ${PORT}
ServerAdmin "${HTTPD_SERVER_ADMIN}"
ServerName "0.0.0.0"
DocumentRoot "${HOME}/#{WEBDIR}"
Include conf/extra/httpd-modules.conf
Include conf/extra/httpd-directories.conf
Include conf/extra/httpd-mime.conf
Include conf/extra/httpd-logging.conf
Include conf/extra/httpd-mpm.conf
Include conf/extra/httpd-default.conf
Include conf/extra/httpd-remoteip.conf
Include conf/extra/httpd-php.conf
Include conf/extra/httpd-my-special-config.conf # This line includes your additional file.
```


For example, adding a file `.bp-config/php/php.ini.d/something.ini` to your app, with the following contents, overrides both the default charset and mimetype:

```
default_charset="UTF-8"
default_mimetype="text/html"
```

**Precedence**

In order of highest precedence, php configuration values come from the following sources:

- php scripts using `ini_set()` to manually override config files
- user.ini files for local values
- `.bp-config/php/php.ini.d` to override master value, but not local values from user.ini files
- `.bp-config/php/fpm.d/`

The buildpack adds any files it finds in the application under `.bp-config/php/fpm.d/` that end with `.conf` (i.e. `my-config.conf`) to the PHP-FPM configuration. You can use this to change any value acceptable to `php-fpm.conf`. See [http://php.net/manual/en/install.fpm.configuration.php](http://php.net/manual/en/install.fpm.configuration.php) for a list of directives.

PHP FPM config snippets are included by the buildpack into the global section of the configuration file. If you need to apply configuration settings for a PHP FPM worker, that needs to be indicated in your configuration file as well.
For example:

```ini
; This option is specific to the 'www' pool
[www]
catch_workers_output = yes
```

### PHP Extensions

The buildpack adds any `.bp-config/php/php.ini.d/FILE-NAME.ini` files it finds in the application to the PHP configuration. You can use this to enable PHP or ZEND extensions. For example:

```ini
extension=redis.so
extension=gd.so
zend_extension=opcache.so
```

If an extension is already present and enabled in the compiled php, for example `intl`, you do not need to explicitly enable it to use that extension.

### PHP_EXTENSIONS vs. ZEND_EXTENSIONS

PHP has two kinds of extensions, PHP extensions and Zend extensions. These hook into the PHP executable in different ways. See [https://wiki.php.net/internals/extensions](https://wiki.php.net/internals/extensions) for more information about the way extensions work internally in the engine.

Because they hook into the PHP executable in different ways, they are specified differently in ini files. Apps fail if a Zend extension is specified as a PHP extension, or a PHP extension is specified as a Zend extension.

If you see the following error, move the example `extension` from `extension` to `zend_extension`, then re-push your app:

```plaintext
php-fpm | [warn-ioncube] The example Loader is a Zend-Engine extension and not a module (pid 40)
php-fpm | [warn-ioncube] Please specify the Loader using 'zend_extension' in php.ini (pid 40)
php-fpm | NOTICE: PHP message: PHP Fatal error: Unable to start example Loader module in Unknown on line 0
```

If you see the following error, move the example `extension` from `zend_extension` to `extension`, then re-push your app.

```plaintext
NOTICE: PHP message: PHP Warning: example MUST be loaded as a Zend extension in Unknown on line 0
```

### PHP Modules

You can include the following modules by adding them to the `PHP_MODULES` list:

- `cli`, installs `php` and `phar`
- `fpm`, installs `PHP-FPM`
- `cgi`, installs `php-cgi`
- `pear`, installs `Pear`

By default, the buildpack installs the `cli` module when you push a standalone app, and installs the `fpm` module when you push a web app. You must specify `cgi` and `pear` if you want them installed.

### Buildpack Extensions

The buildpack comes with extensions for its default behavior. These are the HTTPD, Nginx, PHP, and NewRelic extensions.

The buildpack is designed with an extension mechanism, allowing app developers to add behavior to the buildpack without modifying the buildpack code.

When you push an app, the buildpack runs any extensions found in the `extensions` directory of your app.
The Developer Documentation explains how to write extensions.
Composer

Composer is activated when you supply a composer.json or composer.lock file. A composer.lock is not required, but is strongly recommended for consistent deployments.

You can require dependencies for packages and extensions. Extensions must be prefixed with the standard ext-. If you reference an extension that is available to the buildpack, it will automatically be installed. See the main README for a list of supported extensions.

The buildpack uses the version of PHP specified in your composer.json or composer.lock file. Composer settings override the version set in the options.json file.

The PHP buildpack supports a subset of the version formats supported by Composer. The buildpack supported formats are:

<table>
<thead>
<tr>
<th>Example</th>
<th>Expected Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;5.3</td>
<td>latest 5.4.x release (5.3 is not supported)</td>
</tr>
<tr>
<td>=5.5</td>
<td>latest 5.4.x release</td>
</tr>
<tr>
<td>&gt;5.5</td>
<td>latest 5.5.x release</td>
</tr>
<tr>
<td>&gt;5.4</td>
<td>specific 5.4.x release that is listed</td>
</tr>
<tr>
<td>5.4.x</td>
<td>specific 5.4.x release that is listed</td>
</tr>
</tbody>
</table>

Configuration

The buildpack runs with a set of default values for Composer. You can adjust these values by adding a .bp-config/options.json file to your application and setting any of the following values in it.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPOSER_VERSION</td>
<td>The version of Composer to use. It defaults to the latest bundled with the buildpack.</td>
</tr>
<tr>
<td>COMPOSER_INSTALL_OPTIONS</td>
<td>A list of options that should be passed to composer install. This defaults to --no-interaction --no-dev --no-progress. The --no-progress option must be used due to the way the buildpack calls Composer.</td>
</tr>
<tr>
<td>COMPOSER_VENDOR_DIR</td>
<td>Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to create the vendor directory. Defaults to {BUILD_DIR}/{LIBDIR}/vendor.</td>
</tr>
<tr>
<td>COMPOSER_BIN_DIR</td>
<td>Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to place executables from packages. Defaults to {BUILD_DIR}/php/bin.</td>
</tr>
<tr>
<td>COMPOSER_CACHE_DIR</td>
<td>Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to place its cache files. Generally you should not change this value. The default is {CACHE_DIR}/composer which is a subdirectory of the cache folder passed in to the buildpack. Composer cache files will be restored on subsequent application pushes.</td>
</tr>
</tbody>
</table>

By default, the PHP buildpack uses the composer.json and composer.lock files that reside inside the root directory, or in the directory specified as WEBDIR in your options.json. If you have composer files inside your app, but not in the default directories, use a COMPOSER_PATH environment variable for your app to specify this custom location, relative to the app root directory. Note that the composer.json and composer.lock files must be in the same directory.

GitHub API Request Limits

Composer uses GitHub's API to retrieve zip files for installation into the application folder. If you do not vendor dependencies before pushing an app, Composer will fetch dependencies during staging using the GitHub API.

GitHub's API is request-limited. If you reach your daily allowance of API requests (typically 60), GitHub's API returns a 403 error and staging fails.
There are two ways to avoid the request limit:

1. Vendor dependencies before pushing your application.
2. Supply a GitHub OAuth API token.

Vendor Dependencies

To vendor dependencies, you must run `composer install` before you push your application. You might also need to configure `COMPOSER_VENDOR_DIR` to "vendor".

Supply a GitHub Token

Composer can use GitHub API OAuth tokens, which increase your request limit, typically to 5000 per day.

During staging, the buildpack looks for this token in the environment variable `COMPOSER_GITHUB_OAUTH_TOKEN`. If you supply a valid token, Composer uses it. This mechanism does not work if the token is invalid.

To supply the token, you can use either of the following methods:

1. `cf set-env YOUR_APP_NAME COMPOSER_GITHUB_OAUTH_TOKEN "OAUTH_TOKEN_VALUE"`
2. Add the token to the `env` block of your application manifest.

Buildpack Staging Environment

Composer runs in the buildpack staging environment. Variables set with `cf set-env` or with a `manifest.yml 'env' block` are visible to Composer.

For example:

```bash
$ cf push a_symfony_app --no-start
$ cf set-env a_symfony_app SYMFONY_ENV "prod"
$ cf start a_symfony_app
```

In this example, `a_symfony_app` is supplied with an environment variable, `SYMFONY_ENV`, which is visible to Composer and any scripts started by Composer.

Non-configurable Environment Variables

User-assigned environment variables are applied to staging and runtime. Unfortunately, `LD_LIBRARY_PATH` and `PHPRC` must be different for staging and runtime. The buildpack takes care of setting these variables, which means user values for these variables are ignored.
Sessions

Page last updated:

Usage

When your application has one instance, it’s mostly safe to use the default session storage, which is the local file system. You would only see problems if your single instance crashes as the local file system would go away and you’d lose your sessions. For many applications, this will work just fine but please consider how this will impact your application.

If you have multiple application instances or you need a more robust solution for your application, then you’ll want to use Redis or Memcached as a backing store for your session data. The build pack supports both and when one is bound to your application it will detect it and automatically configure PHP to use it for session storage.

By default, there’s no configuration necessary. Create a Redis or Memcached service, make sure the service name contains `redis-sessions` or `memcached-sessions` and then bind the service to the application.

Example:

```
$ cf create-service redis some-plan app-redis-sessions
$ cf bind-service app app-redis-sessions
$ cf restage app
```

If you want to use a specific service instance or change the search key, you can do that by setting either `REDIS_SESSION_STORE_SERVICE_NAME` or `MEMCACHED_SESSION_STORE_SERVICE_NAME` in `.bp-config/options.json` to the new search key. The session configuration extension will then search the bound services by name for the new session key.

Configuration Changes

When detected, the following changes will be made.

Redis

- the `redis` PHP extension will be installed, which provides the session save handler
- `session.name` will be set to `PHPSESSIONID` which disables sticky sessions
- `session.save_handler` is configured to `redis`
- `session.save_path` is configured based on the bound credentials, for example `tcp://host:port?auth=pass`

Memcached

- the `memcached` PHP extension will be installed, which provides the session save handler
- `session.name` will be set to `PHPSESSIONID` which disables sticky sessions
- `session.save_handler` is configured to `memcached`
- `session.save_path` is configured based on the bound credentials (i.e. `PERSISTENT=app_sessions host:port`)
- `memcached.sess_binary` is set to `On`
- `memcached.use_sasl` is set to `On`, which enables authentication
- `memcached.sess_sasl_username` and `memcached.sess_sasl_password` are set with the service credentials
New Relic

You can configure New Relic for the PHP buildpack in one of two ways:

- With a license key
- With a Cloud Foundry service

With a License Key

Use this method if you already have a New Relic account,

1. In a web browser, navigate to the New Relic website to find your license key.
2. Set the value of the environment variable `NEWRELIC_LICENSE` to your New Relic license key, either through the `manifest.yml` file or by running the `cf set-env` command.

For more information, see [https://github.com/cloudfoundry/php-buildpack#supported-software](https://github.com/cloudfoundry/php-buildpack#supported-software)

With a Cloud Foundry Service

To configure New Relic for the PHP buildpack with a Cloud Foundry service, bind a New Relic service to the app. The buildpack automatically detects and configures New Relic.

Your `VCAP_SERVICES` environment variable must contain a service named `newrelic`, the `newrelic` service must contain a key named `credentials`, and the `credentials` key must contain a named `licenseKey`.

NOTE: You cannot configure New Relic for the PHP buildpack with user-provided services.
Push an App

Cloud Foundry automatically uses this buildpack if it detects a `requirements.txt` or `setup.py` file in the root directory of your project.

If your Cloud Foundry deployment does not have the Python Buildpack installed, or the installed version is out of date, you can use the latest version by specifying it with the `-b` option when you push your app. For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/python-buildpack.git
```

Supported Versions

You can find the list of supported Python versions in the [Python buildpack release notes](https://github.com/cloudfoundry/python-buildpack).

Specify a Python Version

You can specify a version of the Python runtime by including it within a `runtime.txt` file. For example:

```
$ cat runtime.txt
python-3.5.2
```

The buildpack only supports the stable Python versions, which are listed in the `manifest.yml` and [Python buildpack release notes](https://github.com/cloudfoundry/python-buildpack).

To request the latest Python version in a patch line, replace the patch version with `x`: `3.6.x`. To request the latest version in a minor line, replace the minor version: `3.x`.

If you try to use a binary that is not currently supported, staging your app fails and you see the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST: ...
? ? exit
? ?
Staging failed: Buildpack compilation step failed
```

Specify a Start Command

The Python buildpack does not generate a default start command for your applications.

To stage with the Python buildpack and start an application, do one of the following:

- Supply a Proclfile. For more information about Proclfiles, see the [Configuring a Production Server](https://docs.cloudfoundry.orgicap) topic. The following example Proclfile specifies `gunicorn` as the start command for a web app running on Gunicorn:

  ```
  web: gunicorn SERVER-NAME:APP-NAME
  ```

- Specify a start command with `-c`. The following example specifies `waitress-serve` as the start command for a web app running on Waitress:

  ```
  $ cf push python-app -c "waitress-serve --port=$PORT DJANGO-WEB-APP.wsgi:MY-APP"
  ```

- Specify a start command in the application manifest by setting the `command` attribute. For more information, see the [Deploying with Application Manifests](https://docs.cloudfoundry.orgicap) topic.
Vendor App Dependencies

If you are deploying in an environment that is disconnected from the Internet, your application must vendor its dependencies.

For the Python buildpack, use `pip`:

```
$ cd YOUR-APP-DIR
$ mkdir -p vendor
$ vendors all the pip *.tar.gz into vendor/
$ pip download -r requirements.txt --no-binary=:all: -d vendor

cf push uploads your vendored dependencies. The buildpack installs them directly from the `vendor` directory.
```

**Note:** To ensure proper installation of dependencies, we recommend non-binary vendored dependencies. The above `pip install` command achieves this.

Private Dependency Repository

To deploy apps in an environment that needs to use a private dependency repository, you have the following options:

- **PIP**
- **Conda**

**PIP**

To install dependencies using PIP, add the URL of the repository to the `requirements.txt` file in the following format:

```
--index-url=https://example.com/api/pypi/ext_pypi/simple
fixtures==2.0.0
```

If the private repository uses a custom SSL certificate that is installed on the platform, you may see an error similar to the following:

```
Could not fetch URL https://example.com/api/pypi/ext_pypi/simple/fixtures/:
There was a problem confirming the ssl certificate:
[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777) - skipping
```

This error occurs because `pip` does not use system certificates by default. To resolve this issue, set the `PIP_CERT` environment variable in the `manifest.yml` file to point to the system certificate store.

For example:

```
---
env:
  PIP_CERT: /etc/ssl/certs/ca-certificates.crt
```

**Conda**

To install dependencies using Conda, add a `channels` block to the `environment.yml` file.

In the `channels` block, list custom channels and add `nodefaults`. Specifying `nodefaults` tells Conda to only use the channels in the channels block.

For example:

```
channels:
  - https://conda.example.com/repo
  - nodefaults
```

If the private repository uses a custom SSL certificate that is installed on the platform, you may see an error similar to the following:
This error occurs because conda does not use system certificates by default. To resolve this issue, set the CONDA_SSL_VERIFY environment variable in the manifest.yml file to point to the system certificate store.

For example:

```yaml
---
env:
  CONDA_SSL_VERIFY: /etc/ssl/certs/ca-certificates.crt
```

### Parse Environment Variables

The cfenv package provides access to Cloud Foundry application environment settings by parsing all the relevant environment variables. The settings are returned as a class instance. See [https://github.com/jmcarp/py-cfenv](https://github.com/jmcarp/py-cfenv) for more information.

### Miniconda Support

(Starting in buildpack version 1.5.6)

To use miniconda instead of pip for installing dependencies, place an environment.yml file in the root directory.

For examples, see our sample apps:

- [Using Python 2 with miniconda](#)
- [Using Python 3 with miniconda](#)

### Pipenv Support

(Starting in buildpack version 1.5.19)

To use Pipenv instead of pip (directly) for installing dependencies, place a Pipfile in the root directory. Easiest to let Pipenv generate this for you.

### NLTK Support

To use NLTK corpora in your app, you can include an nltk.txt file in the root of your application. Each line in the file specifies one dataset to download. The full list of data sets available this way can be found [on the NLTK website](#). The id listed for the corpora on that page is the string you should include in your app's nltk.txt.

Example nltk.txt:

```
  brown
  wordnet
```

Having an nltk.txt file only causes the buildpack to download the corpora. You still must specify NLTK as a dependency of your app if you want to use it to process the corpora files.

### Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the http_proxy and https_proxy environment variables. For more information, see the [Proxy Usage Documentation](#).

### BOSH Configured Custom Trusted Certificate Support

Versions of Python 2.7.9 and later use certificates stored in /etc/ssl/certs. Your platform operator can configure the platform to [add the custom certificates into the application container](#).
Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Python buildpack in Cloud Foundry, see the python-buildpack GitHub repository.

You can find current information about this buildpack on the Python buildpack release page in GitHub.
R Buildpack

Page last updated:

This topic describes how to push your R app to Cloud Foundry and how to configure your R app to use the R buildpack.

Push an App

Cloud Foundry automatically uses the R buildpack if it detects a `r.yml` file in the root directory of your project.

If your Cloud Foundry deployment does not have the R buildpack installed, or the installed version is out of date, you can use the latest version by specifying it with the `-b` option when you push your app. For example:

```bash
$ cf push my_app -b https://github.com/cloudfoundry/r-buildpack.git
```

Supported Versions

You can find the list of supported R versions in the R buildpack release notes.

Start Command

The R buildpack does not generate a default start command for your applications. Instead, you must specify a start command for your app.

To stage an app with the R buildpack and start the app, do one of the following:

- **Option 1:** Supply a Procfile. For more information about Procfiles, see Production Server Configuration. The following example Procfile specifies `R-f` as the start command for a web app with the entrypoint `simple.r`:

  ```
  web: R -f simple.r
  ```

- **Option 2:** Specify a start command with `-c`:

  ```bash
  $ cf push r-app -c "R -f simple.r"
  ```

- **Option 3:** Specify a start command in the application manifest by setting the `command` attribute. For more information, see the Deploying with Application Manifests topic.

For more information about starting apps, see Starting, Restarting, and Restaging Applications.

Specifying App Dependencies

As of v0.0.5, the following packages are provided by the buildpack:

- Rserve
- forecast
- shiny

To specify additional packages needed by your app, provide the CRAN mirror and names of the packages inside your `r.yml` file. For example:

```yaml
---
packages:
  - cran_mirror: https://cran.r-project.org
  packages:
    - name: string
    - name: jsonlite
```
Vendoring App Dependencies

If you are deploying in an environment that is disconnected from the Internet, you must vendor your app's dependencies, which means you must make these packages available for offline use. You can vendor dependencies by using a package manager.

To set up your own custom, local CRAN-like repository to vendor your packages, create the `src/contrib` directories and populate them with your package source tarballs. For more information, see How to Set Up a Custom CRAN-like Repository published on the Packrat Documentation site.

Add the `src/contrib` directories containing your package tarballs to a `vendor_r` directory at the root of your app. This directory is named `vendor_r` so as not to conflict with vendor directories of other languages, such as python.

Then, inside your `r.yml`, provide the names of your vendored packages in the `packages` list:

```yaml
packages:
  - packages:
    - name: stringr
    - name: jsonlite
```

`cf push` uploads your vendored dependencies. The buildpack installs them directly from the `vendor_r/` directory.

Example app directory tree:

```
├── r.yml
├── simple-app.r
└── vendor_r
    └── src
        └── contrib
            ├── PACKAGES
            └── jsonlite_1.5.tar.gz
    └── src/contrib
        └── stringr_1.3.1.tar.gz
```

For more information about using buildpacks in disconnected environments, see Disconnected environments.

Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH Configured Custom Trusted Certificate Support

R uses certificates stored in `etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container. For more information, see Configuring Trusted System Certificates for Applications.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the R buildpack in Cloud Foundry, see the R-buildpack GitHub repository.

You can also find current information about this buildpack on the R buildpack release page in GitHub.
Ruby Buildpack

Page last updated:

Push Apps

Cloud Foundry uses the Ruby buildpack if your app has a Gemfile and Gemfile.lock in the root directory. The Ruby buildpack uses Bundler to install your dependencies.

If your Cloud Foundry deployment does not have the Ruby buildpack installed or the installed version is out of date, push your app with the -b option to specify the buildpack:

```
cf push MY-APP -b https://github.com/cloudfoundry/ruby-buildpack.git
```

For more detailed information about deploying Ruby applications see the following topics:

- Tips for Ruby Developers
- Getting Started Deploying Apps
- Configuring Rake Tasks for Deployed Apps
- Environment Variables Defined by the Ruby Buildpack
- Configuring Service Connections for Ruby
- Support for Windows Gemfiles

You can find the source for the Ruby buildpack in the Ruby buildpack repository on GitHub.

Supported Versions

You can find supported Ruby versions in the release notes for the Ruby buildpack on GitHub.

Specify a Ruby Version

Specify specific versions of the Ruby runtime in the Gemfile for your app as described in the sections below.

Note: If you don’t specify a version of the Ruby runtime, we will use the default Ruby version listed in the buildpack manifest.yml file.

MRI

For MRI, specify the version of Ruby in your Gemfile as follows:

```
ruby '~> 2.2.3'
```

With this example declaration in the Gemfile, if Ruby versions 2.2.4, 2.2.5, and 2.3.0 are present in the Ruby buildpack, the app uses Ruby version 2.2.5.

For more information about the ruby directive for Bundler Gemfiles, see the Bundler documentation.

Note: If you use v1.6.18 or earlier, you must specify an exact version, such as ruby "2.2.3". In Ruby Buildpack v1.6.18 and earlier, Rubygems do not support version operators for the ruby directive.

JRuby

© Copyright Pivotal Software Inc, 2013-2019
For JRuby, specify the version of Ruby in your `Gemfile` based on the version of JRuby your app uses.

**JRuby version 1.7.x** supports either **1.9** mode or **2.0** mode.

- For **1.9** mode, use the following:
  
  ```ruby
  ruby '1.9.3', :engine => 'jruby', :engine_version => '1.7.25'
  ```

- For **2.0** mode, use the following:
  
  ```ruby
  ruby '2.0.0', :engine => 'jruby', :engine_version => '1.7.25'
  ```

For **Jruby version >= 9.0**, use the following:

```ruby
ruby '2.2.3', :engine => 'jruby', :engine_version => '9.0.5.0'
```

The Ruby buildpack only supports the stable Ruby versions listed in the `manifest.yml` and release notes for the Ruby buildpack on GitHub.

If you try to use a binary that is not supported, staging your app fails with the following error message:

```bash
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST: ...

! ? exit
!

Staging failed: Buildpack compilation step failed
```

**Note:** The Ruby buildpack does not support the pessimistic version operator `~>` on the Gemfile `ruby` directive for JRuby.

### Vendor App Dependencies

As stated in the [Disconnected Environments documentation](#), your application must ‘vendor’ its dependencies.

For the Ruby buildpack, use the `bundle package --all` command in Bundler to vendor the dependencies.

**Example:**

```bash
$ cd my-app-directory
$ bundle package --all
```

The `cf push` command uploads your vendored dependencies. The Ruby buildpack compiles any dependencies requiring compilation while staging your app.

### Buildpack Logging and Application Logging

The Ruby buildpack only runs during the staging process, and only logs what is important to staging, such as what is being downloaded, what the configuration is, and work that the buildpack does on your application.

The buildpack stops logging when the staging process finishes. The Loggregator handles application logging.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. For more information, see the [Application Logging in Cloud Foundry](#) topic.

If you are deploying a Rails application, the buildpack may or may not automatically install the necessary plugin or gem for logging, depending on the Rails version of the application:

- **Rails 2.x:** The buildpack automatically installs the `rails_log_stdout` plugin into the application. For more information about the `rails_log_stdout` plugin, refer to the [GitHub README](#).

- **Rails 3.x:** The buildpack automatically installs the `rails_12factor` gem if it is not present and issues a warning message. You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message. For more information about the `rails_12factor` gem, refer to the [GitHub README](#).

- **Rails 4.x:** The buildpack only issues a warning message that the `rails_12factor` gem is not present, but does not install the gem. You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message. For more information about the `rails_12factor` gem, refer to the [GitHub README](#).

For more information about the `rails_12factor` gem, refer to the [GitHub README](#).
Proxy Support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH Configured Custom Trusted Certificate Support

Ruby uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the app container.

Help and Support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Ruby buildpack in Cloud Foundry, see the Ruby buildpack repository on GitHub.

You can find current information about the Ruby buildpack in the release notes for the Ruby buildpack on GitHub.
Tips for Ruby Developers

Page last updated:

This page has information specific to deploying Rack, Rails, or Sinatra apps.

App Bundling

You must run `Bundler` to create a `Gemfile` and a `Gemfile.lock`. These files must be in your app before you push to Cloud Foundry.

Rack Config File

For Rack and Sinatra, you must have a `config.ru` file. For example:

```ruby
require './hello_world'
run HelloWorld.new
```

Asset Precompilation

Cloud Foundry supports the Rails asset pipeline. If you do not precompile assets before deploying your app, Cloud Foundry precompiles them when staging the app. Precompiling before deploying reduces the time it takes to stage an app.

Use the following command to precompile assets before deployment:

```
rake assets:precompile
```

Note that the Rake precompile task reinitializes the Rails app. This could pose a problem if initialization requires service connections or environment checks that are unavailable during staging. To prevent reinitialization during precompilation, add the following line to `application.rb`:

```ruby
config.assets.initialize_on_precompile = false
```

If the `assets:precompile` task fails, Cloud Foundry uses live compilation mode, the alternative to asset precompilation. In this mode, assets are compiled when they are loaded for the first time. You can force live compilation by adding the following line to `application.rb`:

```ruby
Rails.application.config.assets.compile = true
```

Running Rake Tasks

Cloud Foundry does not provide a mechanism for running a Rake task on a deployed app. If you need to run a Rake task that must be performed in the Cloud Foundry environment, rather than locally before deploying or redeploying, you can configure the command that Cloud Foundry uses to start the app to invoke the Rake task.

An app start command is configured in the app manifest file, `manifest.yml`, using the `command` attribute.

For more information about app manifests and supported attributes, see the Deploying with Application Manifests topic.

Example: Invoking a Rake database migration task at app startup

The following is an example of migrating a database schema using a Rake task. For more information about migrating database schemas, see Services Overview.

```ruby
1.  If a Rakefile does not exist, create one and add it to your app directory.
```
2. In your Rakefile, add a Rake task to limit an idempotent command to the first instance of a deployed app:

```ruby
namespace :cf do
  desc "Only run on the first application instance"
  task :on_first_instance do
    instance_index = JSON.parse(ENV['VCAP_APPLICATION'])['instance_index'] rescue nil
    exit(0) unless instance_index == 0
  end
end
```

3. Add the task to the `manifest.yml` file, referencing the idempotent command `rake db:migrate` with the `command` attribute.

```yaml
---
applications:
- name: my-rails-app
  command: bundle exec rake cf:on_first_instance db:migrate && bundle exec rails s -p $PORT -e $RAILS_ENV
```

4. Update the app using `cf push`.

Rails 3 Worker Tasks

This section shows you how to create and deploy an example Rails app that uses a worker library to defer a task that a separate app executes.

The guide also describes how to scale the resources available to the worker app.

Note: Most worker tasks do not serve external requests. Use the `--no-route` flag with the `cf push` command, or `no-route: true` in the app manifest, to suppress route creation and remove existing routes.

Choose a Worker Task Library

You must choose a worker task library. The table below summarizes the three main libraries available for Ruby / Rails:

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delayed::Job</td>
<td>A direct extraction from Shopify where the job table is responsible for a multitude of core tasks.</td>
</tr>
<tr>
<td>Resque</td>
<td>A Redis-backed library for creating background jobs, placing those jobs on multiple queues, and processing them later.</td>
</tr>
<tr>
<td>Sidekiq</td>
<td>Uses threads to handle many messages at the same time in the same process. It does not require Rails, but integrates tightly with Rails 3 to simplify background message processing. This library is Redis-backed and semi-compatible with Resque messaging.</td>
</tr>
</tbody>
</table>

For other alternatives, see https://www.ruby-toolbox.com/categories/Background_Jobs.

Create an Example App

For the purposes of the example app, we use Sidekiq.

First, create a Rails app with an arbitrary model named "Things":

```
rails create rails-sidekiq
```

```
cd rails-sidekiq
rails g model Thing title:string description:string
```

Add `sidekiq` and `uuidtools` to the Gemfile:
source https://rubygems.org'  
gem 'rails', '3.2.9'  
gem 'mysql2'  
group 'assets' do  
gem 'sass-rails', '~> 3.2.3'  
gem 'coffee-rails', '~> 3.2.1'  
gem 'uglifier', '~> 1.0.3'  
end  
gem 'jquery-rails'  
gem 'sidekiq'  
gem 'uuidtools'  

Install the bundle.  

$ bundle install  

In `app/workers`, create a worker for Sidekiq to carry out its tasks:  

$ touch app/workers/thing_worker.rb  

class ThingWorker  
  include Sidekiq::Worker  
  def perform(count)  
    count.times do  
      thing_uuid = UUIDTools::UUID.random_create.to_s  
      Thing.create :title => 'New Thing (#{thing_uuid})', :description =>  
        'Description for thing #{thing_uuid}'  
    end  
  end  
end  

This worker creates `n` number of things, where `n` is the value passed to the worker.  

Create a controller for "Things":  

$ rails g controller Thing  

class ThingController < ApplicationController  
  def new  
    ThingWorker.perform_async(2)  
    redirect_to '/thing'  
  end  
  def index  
    @things = Thing.all  
  end  
end  

Add a view to inspect our collection of "Things":  

$ mkdir app/views/things  
$ touch app/views/things/index.html.erb  

Deploy the App  

This app needs to be deployed twice for it to work, once as a Rails web app and once as a standalone Ruby app. One way to do this is to keep separate
Cloud Foundry manifests for each app type:

**Web Manifest:** Save this as `web-manifest.yml`:

```
---
applications:
- name: sidekiq
  memory: 256M
  instances: 1
  host: sidekiq
  domain: $[target-base]
  path: .
  services:
    - sidekiq-mysql:
    - sidekiq-redis:
```  

**Worker Manifest:** Save this as `worker-manifest.yml`:

```
---
applications:
- name: sidekiq-worker
  memory: 256M
  instances: 1
  path:
    command: bundle exec sidekiq
  no-route: true
  services:
    - sidekiq-redis:
    - sidekiq-mysql:
```  

Since the url “sidekiq.cloudfoundry.com” is probably already taken, change it in `web-manifest.yml` first, then push the app with both manifest files:

```
$ cf push -f web-manifest.yml
$ cf push -f worker-manifest.yml
```

If the cf CLI asks for a URL for the worker app, select `none`.

**Test the App**

Test the app by visiting the new action on the “Thing” controller at the assigned url. In this example, the URL would be: [http://sidekiq.cloudfoundry.com/thing/new](http://sidekiq.cloudfoundry.com/thing/new)

This creates a new Sidekiq job which is queued in Redis, then picked up by the worker app. The browser is then redirected to `/thing` which shows the collection of “Things”.

**Scale Workers**

Use the `cf scale` command to change the number of Sidekiq workers.

Example:

```
$ cf scale sidekiq-worker +2
```

**Use rails_serve_static_assets on Rails 4**

By default Rails 4 returns a 404 if an asset is not handled via an external proxy such as Nginx. The rails_serve_static_assets gem enables your Rails server to deliver static assets directly, instead of returning a 404. You can use this capability to populate an edge cache CDN or serve files directly from your web app. The gem enables this behavior by setting the config.serve_static_assets option to `true`, so you do not need to configure it manually.

**Add Custom Libraries**
If your app requires external shared libraries that are not provided by the rootfs or the buildpack, you must place the libraries in an `/ld_library_path` directory at the app root.

**Note:** You must keep these libraries up-to-date. They do not update automatically.

The Ruby buildpack automatically adds the directory `<app-root>/ld_library_path` to `LD_LIBRARY_PATH` so that your app can access these libraries at runtime.

### Environment Variables

You can access environment variables programmatically. For example, you can obtain `VCAP_SERVICES` as follows:

```
ENV['VCAP_SERVICES']
```

Environment variables available to you include both those defined by the system and those defined by the Ruby buildpack, as described below. For more information about system environment variables, see the Application-Specific System Variables section of the Cloud Foundry Environment Variables topic.

#### BUNDLE_BIN_PATH

Location where Bundler installs binaries.

Example: `BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle`

#### BUNDLE_GEMFILE

Path to app Gemfile.

Example: `BUNDLE_GEMFILE:/home/vcap/app/Gemfile`

#### BUNDLE_WITHOUT

The `BUNDLE_WITHOUT` environment variable instructs Cloud Foundry to skip gem installation in excluded groups.

Use this with Rails applications, where “assets” and “development” gem groups typically contain gems that are not needed when the app runs in production.

Example: `BUNDLE_WITHOUT=assets`

#### DATABASE_URL

Cloud Foundry examines the `database_uri` for bound services to see if they match known database types. If known relational database services are bound to the app, the `DATABASE_URL` environment variable is set using the first match in the list.

If your app depends on `DATABASE_URL` to be set to the connection string for your service and Cloud Foundry does not set it, use the `cf set-env` command to set this variable manually.

Example:

```
$ cf set-env my-app-name DATABASE_URL mysql://example-database-connection-string
```

#### GEM_HOME

Location where gems are installed.
Example: `GEM_HOME=/home/vcap/app/vendor/bundle/ruby/1.9.1`

GEM_PATH
Location where gems can be found.
Example: `GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:

RACK_ENV
This variable specifies the Rack deployment environment. Valid values are `development`, `deployment`, and `none`. This governs which middleware is loaded to run the app.
Example: `RACK_ENV=development`

RAILS_ENV
This variable specifies the Rails deployment environment. Valid values are `development`, `test`, and `production`. This controls which of the environment-specific configuration files governs how the app is executed.
Example: `RAILS_ENV=production`

RUBYOPT
This Ruby environment variable defines command-line options passed to Ruby interpreter.
Example: `RUBYOPT=-I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup`
Getting Started Deploying Ruby Apps

Page last updated:

This topic provides links to additional information about getting started deploying apps using the Ruby buildpack. See the following topics for deployment guides specific to your app language and framework:

- Ruby
- Ruby on Rails
Getting Started Deploying Ruby Apps

This guide is intended to walk you through deploying a Ruby app to Pivotal Application Service (PAS). If you experience a problem following the steps below, check the Known Issues topic, or refer to the Troubleshooting Application Deployment and Health topic.

Sample App Step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_ruby.git` to clone the `pong_matcher_ruby` app from GitHub, and follow the instructions in the Sample App Step sections.

**Note:** Ensure that your Ruby app runs locally before continuing with this procedure.

Deploy a Ruby Application

This section describes how to deploy a Ruby application to PAS, and uses output from a sample app to show specific steps of the deployment process.

Prerequisites

- A Ruby 2.x application that runs locally on your workstation
- Bundler configured on your workstation
- Basic to intermediate Ruby knowledge
- The Cloud Foundry Command Line Interface (cf CLI) installed on your workstation

Step 1: Create and Bind a Service Instance for a Ruby Application

This section describes using the cf CLI to configure a Redis Cloud managed service instance for an app. You can use either the CLI or Apps Manager to perform this task.

PAS supports two types of service instances:

- Managed services integrate with PAS through service brokers that offer services and plans and manage the service calls between PAS and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans that are available to you.

The example shows three of the available managed database-as-a-service providers and the plans that they offer: `cleardb` MySQL and `elephantsql` PostgreSQL as a Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK
<table>
<thead>
<tr>
<th>service</th>
<th>plans</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cleardb</td>
<td>spark, boost, amp, shock</td>
<td>Highly available MySQL for Your Apps</td>
</tr>
<tr>
<td>elephantsql</td>
<td>turtle, panda, hippo, elephant</td>
<td>PostgreSQL as a Service</td>
</tr>
</tbody>
</table>
```

© Copyright Pivotal Software Inc, 2013-2019 1449 2.5
Run `cf create-service SERVICE PLAN SERVICE_INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE_INSTANCE.

**Sample App Step**

Run `cf create-service rediscloud 30mb redis`. This creates a service instance named `redis` that uses the `rediscloud` service and the `30mb` plan, as the example below shows.

```
$ cf create-service rediscloud 30mb redis
Creating service redis in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Bind a Service Instance

When you bind an app to a service instance, PAS writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider’s documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App Step.

You can also bind a service using the [Apps Manager](#).

**Sample App Step**

You can skip this step. The manifest for the sample app contains a `services` sub-block in the `applications` block, as the example below shows. This binds the `redis` service instance that you created in the previous step.

```
services:
  - redis
```

**Step 2: Configure Deployment Options**

Configure the Deployment Manifest

You can specify app deployment options in a manifest that the `cf push` command uses. For more information about application manifests and supported attributes, refer to the [Deploying with Application Manifests](#) topic.

Configure a Production Server

PAS uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, PAS can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn. If your app requires a more robust web server, refer to the [Configuring a Production Server](#) topic for help configuring a server other than WEBrick.

**Sample App Step**

You can skip this step. The `manifest.yml` file for `pong_matcher_ruby` does not require any additional configuration to deploy the app.
Step 3: Log in and Target the API Endpoint

Run:
```
  cf login -a API_ENDPOINT
```

enter your login credentials, and select a space and org. The API endpoint is the URL of the Cloud Controller in your PAS instance.

Sample App Step

You must do this step to run the sample app.

Step 4: Deploy an App

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run:
```
  cf push APP_NAME
```

creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP_NAME and DOMAIN is specified by your administrator. Your DOMAIN is shared-domain.example.com. For example:
```
  cf push my-app
```
creates the URL my-app.shared-domain.example.com.

The URL for your app must be unique from other apps that PAS hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words.
- `cf help push` to view other options for this command.

If you want to view log activity while the app deploys, launch a new terminal window and run:
```
  cf logs APP_NAME
```

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

Run:
```
  cf push pong_matcher_ruby -n HOST_NAME
```

Example:
```
  cf push pong_matcher_ruby -n pongmatch-ex12
```

The example below shows the terminal output of deploying the pong_matcher_ruby app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the redis service and follows the instructions in the manifest to start one instance of the app with 256M. After the app starts, the output displays the health and status of the app.

Note: The pong_matcher_ruby app does not include a web interface. To interact with the pong_matcher_ruby app, see the interaction instructions on GitHub: [https://github.com/cloudfoundry-samples/pong_matcher_ruby](https://github.com/cloudfoundry-samples/pong_matcher_ruby).
Step 5: Test a Deployed App

You've deployed an app to PAS!

Use the cf CLI or Apps Manager to review information and administer your app and your PAS account. For example, you could edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. See also Using the Apps Manager.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

Note: You cannot perform certain tasks in the CLI or Apps Manager because these are commands that only a PAS administrator can run. If you are not a PAS administrator, the following message displays for these types of commands:

```
error code: 10003, message: You are not authorized to perform the requested action
```

For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the Getting Started with the Apps Manager topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App Deploy Fails
Even when deploying an app fails, the app might exist on PAS. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or Apps Manager, or you might have to delete the app and redeploy.

Common reasons deploying an app fails include:

- You did not successfully create and bind a needed service instance to the app, such as a PostgreSQL service instance. Refer to Step 2: Create and Bind a Service Instance for a Ruby Application.
- You did not successfully create a unique URL for the app. Refer to the troubleshooting tip App Requires Unique URL.

App Requires Unique URL

PAS requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and PAS cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.
Getting Started Deploying Ruby on Rails Apps

This guide walks you through deploying a Ruby on Rails app to Pivotal Cloud Foundry (PCF).

Prerequisites

In order to deploy a sample Ruby on Rails app, you must have the following:

- A Cloud Foundry deployment or a Pivotal Web Services account
- The Cloud Foundry Command Line Interface
- A Cloud Foundry username and password with Space Developer permissions. See your Org Manager if you require permissions.

Step 1: Clone the App

Run the following terminal command to create a local copy of the cf-sample-app-rails.

```
$ git clone https://github.com/cloudfoundry-samples/cf-sample-app-rails.git
```

The newly created directory contains a `manifest.yml` file, which assists CF with deploying the app. See Deploying with Application Manifests for more information.

Step 2: Log in and Target the API Endpoint

1. Run the following terminal command to log in and target the API endpoint of your deployment. For more information, see the Identifying the API Endpoint for your PAS Instance topic.

```
$ cf login -a YOUR-API-ENDPOINT
```

2. Use your credentials to log in, and to select a Space and Org.

Step 3: Create a Service Instance

Run the following terminal command to create a PostgreSQL service instance for the sample app. Our service instance is `rails-postgres`. It uses the elephantsql service and the turtle plan. For more information about the elephantsql service, see ElephantSQL.

```
$ cf create-service elephantsql turtle rails-postgres
```

Step 4: Deploy the App

Make sure you are in the `cf-sample-app-rails` directory. Run the following terminal command to deploy the app:

```
$ cf push cf-sample-app-rails
```

`cf push cf-sample-app-rails` creates a URL route to your application in the form HOST.DOMAIN. In this example, HOST is cf-sample-app-rails. Administrators specify the DOMAIN. For example, for the DOMAIN `shared-domain.example.com`, running `cf push cf-sample-app-rails shared-domain.example.com` creates the URL `cf-sample-app-rails.shared-domain.example.com`.
The example below shows the terminal output when deploying the `cf-sample-app-rails` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then follows the information in the manifest to start one instance of the app with 256M of RAM. After the app starts, the output displays the health and status of the app.

```
$ cf push cf-sample-app-rails
Using manifest file ~/.workspace/cf-sample-app-rails/manifest.yml

Creating app cf-sample-app-rails in org my-org / space dev as clouduser@example.com...
OK

Creating route cf-sample-app-rails.cfapps.io...
OK

Binding cf-sample-app-rails.cfapps.io to cf-sample-app-rails...
OK

Uploading cf-sample-app-rails...
Uploading app files from: ~/.workspace/cf-sample-app-rails
Uploading 746.6K, 136 files
Done uploading
OK

Starting app cf-sample-app-rails in org my-org / space dev as clouduser@example.com...
...
0 of 1 instances running, 1 starting
1 of 1 instances running

App started
OK

App cf-sample-app-rails was started using this command: `bundle exec rails server -p $PORT`

Showing health and status for app cf-sample-app-rails in org my-org / space dev as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: cf-sample-app-rails.cfapps.io
last uploaded: Fri Dec 22 18:08:32 UTC 2017
stack: cflinuxfs3

state since cpu memory disk details
running 2018-8-17 10:09:57 AM 0.0% 20.7M of 512M 186.8M of 1G

TIP: Use `cf restage cf-sample-app-rails` to ensure your env variable changes take effect.
```

**Note:** If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs cf-sample-app-rails`.

### Step 5: Bind the Service Instance

1. Run the command below to bind the service instance to the sample app. Once bound, environment variables are stored that allow the app to connect to the service after a `cf push` , `cf restage` , or `cf restart` command.

```
$ cf bind-service cf-sample-app-rails rails-postgres
Binding service rails-postgres to app cf-sample-app-rails in org my-org / space dev
OK

TIP: Use `cf restage cf-sample-app-rails` to ensure your env variable changes take effect.
```

2. Run the following command to restage the sample app.

```
$ cf restage cf-sample-app-rails
```

3. Run the following command to verify the service instance is bound to the sample app.

```
$ cf services
Getting services in org my-org / space dev
OK

name service plan bound apps last operation
rails-postgres elephantsql turtle cf-sample-app-rails create succeeded
```
Step 6: Verify the App

Verify that the app is running by browsing to the URL generated in the output of the previous step. In this example, navigating to `cf-sample-app-rails.shared-domain.example.com` verifies that the app is running.

You’ve now pushed an app to PAS! For more information about this topic, see the Deploying an Application topic.

What to Do Next

You have deployed an app to PCF. Consult the sections below for information about what to do next.

Test a Deployed App

Use the cf CLI or Apps Manager to review information and administer your app and your PCF account. For example, you could edit the `manifest.yml` file to increase the number of app instances from 1 to 3 or redeploy the app with a new app name.

Manage Your App with the cf CLI

Run `cf help` to view a complete list of commands and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf CLI topics, especially the Getting Started with cf CLI topic.

Troubleshooting

If your app fails to start, verify that the app starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.
Configure Rake Tasks for Deployed Apps

Page last updated:

For PAS to automatically invoke a Rake task while a Ruby or Ruby on Rails app is deployed, you must do the following:

- Include the Rake task in your app
- Configure the application start command using the `command` attribute in the application manifest

The following is an example that shows how to invoke a Rake database migration task at application startup.

1. Create a file with the Rake task name and the extension `.rake`, and store it in the `lib/tasks` directory of your application.

2. Add the following code to your rake file:

```ruby
namespace :cf do
desc "only run on the first application instance"
task :on_first_instance do
  instance_index = JSON.parse(ENV['VCAP_APPLICATION'])['instance_index'] rescue nil
  exit(0) unless instance_index == 0
end
end
```

This Rake task limits an idempotent command to the first instance of a deployed application.

3. Add the task to the `manifest.yml` file with the `command` attribute, referencing the idempotent command `rake db:migrate` chained with a start command.

```yaml
applications:
  - name: my-rails-app
    command: bundle exec rake cf:on_first_instance db:migrate && rails s -p $PORT
```
Environment Variables Defined by the Ruby Buildpack

Pivotal Application Service (PAS) provides configuration information to apps through environment variables. This topic describes the additional environment variables provided by the Ruby buildpack.

For more information about the standard environment variables provided by Pivotal Application Service (PAS), see the Cloud Foundry Environment Variables topic.

Ruby Buildpack Environment Variables

The following table describes the environment variables provided by the Ruby buildpack.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUNDLE_BIN_PATH</td>
<td>The directory where Bundler installs binaries. Example: BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle</td>
</tr>
<tr>
<td>BUNDLE_GEMFILE</td>
<td>The path to the Gemfile for the app. Example: BUNDLE_GEMFILE:/home/vcap/app/Gemfile</td>
</tr>
<tr>
<td>BUNDLE_WITHOUT</td>
<td>Instructs Cloud Foundry to skip gem installation in excluded groups. Use this with Rails applications, where “assets” and “development” gem groups typically contain gems that are not needed when the app runs in production. Example: BUNDLE_WITHOUT=assets</td>
</tr>
</tbody>
</table>
| DATABASE_URL         | Cloud Foundry examines the database_uri for bound services to see if they match known database types. If known relational database services are bound to the app, then the DATABASE_URL environment variable is set to the first services in the list. If your application requires that DATABASE_URL is set to the connection string for your service, and Cloud Foundry does not set it, use the Cloud Foundry Command Line Interface (cf CLI) cf set-env command to set this variable manually. Example: 

```bash
cf set-env my-app DATABASE_URL mysql://example-database-connection-string
```

| GEM_HOME             | The directory where gems are installed. Example: GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1 |
| GEM_PATH             | The directory where gems can be found. Example: GEM_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/ |
| RACK_ENV             | The Rack deployment environment, which governs the middleware loaded to run the app. Valid value are: development, deployment, and none. Example: RACK_ENV=none |
| RAILS_ENV            | The Rails deployment environment, which controls which environment-specific configuration file governs how the app is executed. Valid value are: development, test, and production. Example: RAILS_ENV=production |
| RUBYOPT              | Defines command-line options passed to Ruby interpreter. Example: RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup |
Configure Service Connections for Ruby

After you create a service instance and bind it to an application, you must configure the application to connect to the service.

Query VCAP_SERVICES with cf-app-utils

The `cf-app-utils` gem allows your application to search for credentials from the `VCAP_SERVICES` environment variable by name, tag, or label.

- `cf-app-utils-ruby` 🗄

VCAP_SERVICES defines DATABASE_URL

At runtime, Cloud Foundry creates a `DATABASE_URL` environment variable for every application based on the `VCAP_SERVICES` environment variable.

Example VCAP_SERVICES:

```json
VCAP_SERVICES =
{
  "elephantsql": {
    "name": "elephantsql-c8c6f",
    "label": "elephantsql",
    "credentials": {
      "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb"
    }
  }
}
```

Based on this `VCAP_SERVICES`, Cloud Foundry creates the following `DATABASE_URL` environment variable:

```bash
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb
```

Cloud Foundry uses the structure of the `VCAP_SERVICES` environment variable to populate `DATABASE_URL`. Any service containing a JSON object with the following form will be recognized by Cloud Foundry as a candidate for `DATABASE_URL`:

```json
{
  "some-service": {
    "credentials": {
      "uri": "<some database URL>"
    }
  }
}
```

Cloud Foundry uses the first candidate found to populate `DATABASE_URL`.

Configure Non-Rails Applications

Non-Rails applications can also access the `DATABASE_URL` variable.

If you have more than one service with credentials, only the first will be populated into `DATABASE_URL`. To access other credentials, you can inspect the `VCAP_SERVICES` environment variable.

```bash
vcap_services = JSON.parse(ENV['VCAP_SERVICES'])
```

Use the hash key for the service to obtain the connection credentials from `VCAP_SERVICES`. 
For services that use the v2 format, the hash key is the name of the service.

For services that use the v1 format, the hash key is formed by combining the service provider and version, in the format PROVIDER-VERSION.

For example, the service provider “p-mysql” with version “n/a” forms the hash key p-mysql-n/a.

Seed or Migrate Database

Before you can use your database the first time, you must create and populate or migrate it. For more information about migrating database schemas, see Services Overview.

Troubleshooting

To aid in troubleshooting issues connecting to your service, you can examine the environment variables and log messages Cloud Foundry records for your application.

View Environment Variables

Use the cf env command to view the Cloud Foundry environment variables for your application. cf env displays the following environment variables:

- The VCAP_SERVICES variables existing in the container environment
- The user-provided variables set using the cf set-env command

```bash
$ cf env my-app
Getting env variables for app my-app in org My-Org / space development as admin...
OK

System-Provided:

VCAP_SERVICES:

p-mysql-n/a:

credentials:

uri: postgres://lrra:e6B-X@example.com:5432/lrra

label: p-mysql-n/a

name: p-mysql

syslog_drain_url: ""

tags: ["postgres","postgresql","relational"]

User-Provided:

my-env-var: 100

my-drain: http://drain.example.com
```

View Logs

Use the cf logs command to view the Cloud Foundry log messages for your application. You can direct current logging to standard output, or you can dump the most recent logs to standard output.

Run cf logs APPNAME to direct current logging to standard output:

```
$ cf logs my-app
Connected, tailing logs for app my-app in org My-Org / space development as admin...
```

Run cf logs APPNAME --recent to dump the most recent logs to standard output:

```
Run cf logs my-app --recent
```

© Copyright Pivotal Software Inc, 2013-2019
If you encounter the error, "A fatal error has occurred. Please see the Bundler troubleshooting documentation," update your version of bundler and run:

```
5 gem update bundler
5 gem update --system
5 bundle install
```
Support for Windows Gemfiles

This topic describes how the Ruby buildpack handles dependencies on Windows machines.

Windows Gemfiles

When a Gemfile.lock is generated on a Windows machine, it often contains gems with Windows-specific versions. This results in versions of gems such as mysql2, thin, and pg containing “-x86-mingw32.” For example, the Gemfile may contain the following:

```
gem 'sinatra'
gem 'mysql2'
gem 'json'
```

When you run `bundle install` with the above Gemfile on a Windows machine, it results in the following Gemfile.lock:

```
GEM remote: http://rubygems.org/
specs:
  json (1.7.3)
  mysql2 (0.3.11-x86-mingw32)
  rack (1.4.1)
  rack-protection (1.2.0)
  rack-sinatra (1.3.2)
  rack (~> 1.3, >= 1.3.6)
  rack-protection (~> 1.2)
  tilt (~> 1.3, >= 1.3.1)
  tilt (1.3)
PLATFORMS x86-mingw32
DEPENDENCIES
  json
  mysql2
  sinatra
```

Notice the “-x86-mingw32” in the version number of mysql2. Since Cloud Foundry runs on Linux machines, this would fail to install. To mitigate this, the Ruby Buildpack removes the Gemfile.lock and uses Bundler to resolve dependencies from the Gemfile.

**Note:** Removing the Gemfile.lock will cause dependency versions to be resolved from the Gemfile. This could result in different versions being installed than those listed in the Gemfile.lock.
Overview

This topic describes how to configure and use the Staticfile buildpack.

Note: BOSH configured custom trusted certificates are not supported by the Staticfile buildpack.

Definitions

Staticfile app: An app or content that requires no backend code other than the NGINX webserver, which the buildpack provides. Examples of staticfile apps are front-end JavaScript apps, static HTML content, and HTML/JavaScript forms.

Staticfile buildpack: The buildpack that provides runtime support for staticfile apps and apps with backends hosted elsewhere. To find which version of NGINX the current Staticfile buildpack uses, see the Staticfile buildpack release notes.

Staticfile Detection

If you create a file named Staticfile in the root directory of your app, Cloud Foundry automatically uses the Staticfile buildpack when you push your app.

The Staticfile file can be an empty file, or it can contain configuration settings for your app. For more information, see Configuring the Buildpack and Pushing an App.

Memory Usage

NGINX requires 20 MB of RAM to serve static assets. When using the Staticfile buildpack, we recommend pushing apps with the `-m 64M` option to reduce RAM allocation from the default 1 GB allocated to containers by default.

Configure the Buildpack

This section describes configuration options available for the Staticfile buildpack. If you need to make configuration changes to NGINX that are not listed in the table below, use the NGINX Buildpack instead of the Staticfile buildpack. For more information, see NGINX Buildpack.

Staticfile File Configuration

To configure these options, add the configuration property as a new line in your Staticfile file as described in the following table.

<table>
<thead>
<tr>
<th>Staticfile Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>root: YOUR-DIRECTORY-NAME</td>
<td>Use this option to specify a root directory other than the default. For example, you can specify an alternate folder such as <code>dist</code> or <code>public</code>.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Example: root: public</td>
<td></td>
</tr>
<tr>
<td>directory: visible</td>
<td>Use this option to display an HTML page that shows a directory index for your site. A sample of a directory list is shown below.</td>
</tr>
</tbody>
</table>
If your site is missing an index.html file, your app displays a directory list instead of the standard 404 error page.

**SSI**: Enables Server Side Includes (SSI), which allow you to embed the contents of one or more files into a web page on a web server. For general information about SSI, see the [Server Side Includes](#) entry on Wikipedia.

**pushstate**: enabled

**pushstate**: enabled

**GZip file serving and compression**: Disables gzip_static and gunzip modules, which are enabled by default. These modules allow NGINX to serve files stored in compressed GZ format and to uncompress them for clients that do not support compressed content or responses.

You may want to disable compression under particular circumstances such as serving content to very old browser clients.

**http_proxy**: HTTP-URL

**https_proxy**: HTTPS-URL

Example:

http_proxy: http://www.example.com/
https_proxy: https://www.example.com/

**Proxy support**: Allows you to use a proxy when downloading dependencies during the staging of your app.

**force_https**: true

Alternatively, set the `FORCE_HTTPS` environment variable to `true`.

**host_dot_files**: true

**Dot files**: By default, hidden files, which are those starting with a `.`, are not served by this buildpack.

**status_codes**:

- 404: /404.html
- 500: /500.html

**HTTP Strict Transport Security (HSTS)**: Causes NGINX to respond to all requests with the header:

```
Strict-Transport-Security: max-age=31536000
```

This forces receiving browsers to make all subsequent requests over HTTPS. This setting defaults to a `max-age` of one year.

**HSTS includes subdomains**: Causes NGINX to respond to all requests with the following header:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

This forces browsers to make all subsequent requests over HTTPS including subdomains. This setting defaults to a `max-age` of one year.

**HSTS preload**: Causes NGINX to respond to all requests with the following header:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

This forces browsers to make all subsequent requests over HTTPS including subdomains and requests inclusion in browser-managed HSTS preload lists. For more information, see [https://hstspreload.org](https://hstspreload.org). This setting defaults to a `max-age` of one year. This setting defaults to a `max-age` of one year.

**Note**: Setting this property to `true` also makes `http_strict_transport_security` and `http_strict_transport_security_include_subdomains` default to true.
Other Configuration

Follow the instructions below to make other configuration changes.

Basic Authentication

Allows you to enable basic authentication for your app or website. A sample basic authentication dialog box is shown below:

You can create a hashed username and password pair for each user by using a site like Htpasswd Generator.

Configuration

Add a new Staticfile.auth file to the root or alternative root directory. In the new file, add one or more username and password entries using the following format: 

`USERNAME:HASHED_PASSWORD`

Example:

```
pat:$example1$DuUQEQp8$ZccZCHQElNSjrgerwSFC0
stevie:$example1$22derfaecZSJJRw4rKE$KKEWKSK
```

Custom Location

Allows you to specify custom location definitions with additional directives. For information about NGINX directives, see Creating NGINX Plus and NGINX Configuration Files and Alphabetical index of directives in the NGINX documentation.

Configuration

To customize the location block of the NGINX configuration file, follow the steps below.

1. Set an alternative root directory. The location_include property only works in conjunction with an alternative root.

2. Create a file with location-scoped NGINX directives. See the following example, which causes visitors of your site to receive the X-MySiteName HTTP header:

```
File: nginx/conf/includes/custom_header.conf
Content: add_header X-MySiteName BestSiteEver;
```

3. Set the location_include variable in your Staticfile to the path of the file from the previous step. This path is relative to nginx/conf.

Example:

```
...  
root: public
location_include: includes/*.conf
...  
```

Additional MIME Type Support

Allows you to configure additional MIME types on your NGINX server.

Configuration

To add MIME types, add a mime.types file to your root folder, or to the alternate root folder if you specified one.
For more information about the `mime.types` file, see `mime.types` in the NGINX documentation.

Example MIME types file:

```
  types {
    text/html    html htm shtml;
    text/css     css;
    text/xml     xml rss;
    image/gif    gif;
    ...
  }
```

### Push an App

To push your app, you can use either the system Staticfile buildpack or specify a Staticfile buildpack.

#### Option 1: Use the System Staticfile Buildpack

To use the Staticfile buildpack installed in your deployment, run the command below in the root directory of the app. The root directory of the app must contain a file named `Staticfile`.

```
cf push APP-NAME
```

Where `APP-NAME` is the name you want to give your app.

For example:

```
$ cf push my-app
Creating app my-app in org sample-org / space sample-space as username@example.com...
OK
...
requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: my-app.example.com
stack: cflinuxfs3
buildpack: staticfile

  state since        cpu   memory  disk  details
00 running 2020-04-01 12:55:37 PM 0.0% 12.6M of 1G 5.9M of 1G
```

#### Option 2: Specify a Staticfile Buildpack

To explicitly specify a Staticfile buildpack, run the command below in the root directory of the app. You may want to specify a buildpack if your deployment does not have the Staticfile buildpack installed or the installed version is outdated.

```
cf push APP-NAME -b BUILDPACK-NAME-OR-PATH
```

Where:

- `APP-NAME` is the name you want to give your app.
- `BUILDPACK-NAME-OR-PATH` is either the name of a buildpack that is installed in your deployment or the path to a buildpack. You can find the Cloud Foundry Staticfile buildpack in the [Staticfile repository](https://github.com) on GitHub.

For example:
Verifying the Push

After you push the app, follow the steps below to verify that the push was successful:

1. Find the URL of your app in the output of the `cf push` command. For example, the URL in the terminal output above is `my-app.example.com`.

2. In a browser, navigate to the URL to view your app.

Example Staticfile Buildpack Apps

For different examples of apps that use the Staticfile buildpack, see the fixtures directory in the Staticfile buildpack GitHub repo.

Help and Support

A number of channels exist where you can get more help when using the Staticfile buildpack, or with developing your own Staticfile buildpack.

- **Staticfile Buildpack Repository in GitHub**: Find more information about using and extending the Staticfile buildpack in GitHub repository.
- **Release Notes**: Find current information about this buildpack on the Staticfile buildpack release page in GitHub.
- **Slack**: Join the #buildpacks channel in the Cloud Foundry Slack community.
Customizing and Developing Buildpacks

Buildpacks enable you to packaging frameworks and/or runtime support for your application. Cloud Foundry provides with system buildpacks out-of-the-box and provides an interface for customizing existing buildpacks and developing new ones.

Customizing and Creating Buildpacks

If your application uses a language or framework that the Cloud Foundry system buildpacks do not support, do one of the following:

- Use a Cloud Foundry Community Buildpack.
- Use a Heroku Third-Party Buildpack.
- Customize an existing buildpack or create your own custom buildpack. A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. For information about customizing an existing buildpack or creating your own, see the following:
  - Creating Custom Buildpacks
  - Packaging Dependencies for Offline Buildpacks

Maintaining Buildpacks

After you have modified an existing buildpack or created your own, it is necessary to maintain it. Refer to the following when maintaining your own buildpacks:

- Merging from Upstream Buildpacks
- Upgrading Dependency Versions

Note: To configure a production server for your web app, see the Configuring a Production Server topic.

Using CI for Buildpacks

For information about updating and releasing a new version of a Cloud Foundry buildpack through the Cloud Foundry Buildpacks Team Concourse pipeline, see Using CI for Buildpacks. You can use this as a model when working with Concourse to build and release new versions of your own buildpacks.
Creating Custom Buildpacks

This topic describes how to create custom buildpacks for Pivotal Application Service (PAS).

For more information about how buildpacks work, see the [Buildpacks](#) topic.

Package Custom Buildpacks

PAS buildpacks can work with limited or no Internet connectivity. The `buildpack-packager` gives the same flexibility to custom buildpacks, enabling them to work in partially or completely disconnected environments.

Use the Buildpack Packager

1. Ensure that you have installed the `buildpack-packager`.
2. Create a `manifest.yml` in your buildpack.
3. Run the packager in cached mode:

```
$ buildpack-packager build -cached -any-stack
```

The packager will add (almost) everything in your buildpack directory into a zip file. It will exclude anything marked for exclusion in your manifest.

In cached mode, the packager downloads and adds dependencies as described in the manifest.

For more information, see the documentation in the `buildpack-packager` GitHub repository.

Use and Share the Packaged Buildpack

After you have packaged your buildpack using `buildpack-packager` you can use the resulting `.zip` file locally, or share it with others by uploading it to any network location that is accessible to the CLI. Users can then specify the buildpack with the `-b` option when they push apps. See [Deploying Apps with a Custom Buildpack](#) for details.

**Note:** Offline buildpack packages may contain proprietary dependencies that require distribution licensing or export control measures. For more information about offline buildpacks, refer to [Packaging Dependencies for Offline Buildpacks](#).

You can also use the `cf create-buildpack` command to upload the buildpack into your Cloud Foundry deployment, making it accessible without the `-b` flag:

```
$ cf create-buildpack BUILDPACK PATH POSITION [--enable|--disable]
```

You can find more documentation in the [Managing Custom Buildpacks](#) topic.

Specify a Default Version

As of `buildpack-packager` version 2.3.0, you can specify the default version for a dependency by adding a `default_versions` object to the `manifest.yml` file.

The `default_versions` object has two properties, `name` and `version`. For example:

```
default_versions:
  - name: go
    version: 1.6.3
  - name: other-dependency
    version: 1.1.1
```

To specify a default version:
1. Add the `default_version` object to your manifest, following the rules below. For a complete example, see `manifest.yml` in the go-buildpack repository in GitHub.

2. Run the `default_version_for` script from the `compile-extensions` repository, passing the path of your `manifest.yml` and the dependency name as arguments. The following command uses the example manifest from step 1:

```bash
$ ./compile-extensions/bin/default_version_for manifest.yml go 1.6.3
```

### Rules for Specifying a Default Version

The `buildpack-packager` script validates this object according to the following rules:

- You can create at most one entry under `default_versions` for a single dependency. The following example causes `buildpack-packager` to fail with an error because the manifest specifies two default versions for the same `go` dependency.

```yaml
# Incorrect; will fail to package
default_versions:
  - name: go
    version: 1.6.3
  - name: go
    version: 1.7.5
```

- If you specify a `default_version` for a dependency, you must also list that dependency and version under the `dependencies` section of the manifest. The following example causes `buildpack-packager` to fail with an error because the manifest specifies `version: 1.9.2` for the `go` dependency, but lists `version: 1.7.5` under `dependencies`.

```yaml
# Incorrect; will fail to package
default_versions:
  - name: go
    version: 1.9.2
dependencies:
  - name: go
    version: 1.7.5
    uri: https://storage.googleapis.com/golang/go1.7.5.linux-amd64.tar.gz
    md5: c8cb76e2308c792e2705c2eb1b55de95
cf_stacks:
  - cflinuxfs3
```

### Core Buildpack Communication Contract

This section describes the communication contract followed by the PAS core buildpacks. This contract enables buildpacks to interact with one another, so that developers can use multiple buildpacks with their applications.

Buildpack developers must ensure their custom buildpacks follow the contract.

This section uses the following placeholders:

- `IDX` is the zero-padded index matching the position of the buildpack in the priority list.
- `MD5` is the MD5 checksum of the buildpack's URL.

For all buildpacks that supply dependencies via `/bin/supply`:

- The buildpack must create `/tmp/deps/IDX/config.yml` to provide a name to subsequent buildpacks. This file may also contain miscellaneous configuration for subsequent buildpacks.
- The `config.yml` file should be formatted as follows, replacing `BUILDPACK` with the name of the buildpack providing dependencies and `YAML-OBJECT` with the YAML object that contains buildpack-specific configuration:

```yaml
name: BUILDPACK config: YAML-OBJECT
```
- The following directories may be created inside of `/tmp/deps/IDX/` to provide dependencies to subsequent buildpacks:

```bash
/bin/: Contains binaries intended for SPATH during staging and launch
/lib/: Contains libraries intended for $LD_LIBRARY_PATH during staging and launch
/include/: Contains header files intended for compilation during staging
/pkgconfig/: Contains `.pc` files intended for compilation during staging
/profile.d/: Contains environment vars intended for staging, loaded as: `FILENAME=FILECONTENTS`
/profile.d/: Contains scripts intended for `/app/.profile.d/`, sourced before launch
```

© Copyright Pivotal Software Inc, 2013-2019
The buildpack may make use of previous non-final buildpacks by scanning `/tmp/deps/` for index-named directories containing `config.yml`.

For the last buildpack:

- To make use of dependencies provided by the previously applied buildpacks, the last buildpack must scan `/tmp/deps/` for index-named directories containing `config.yml`.
- To make use of dependencies provided by previous buildpacks, the last buildpack:
  - May use `/bin` during staging, or make it available in $PATH during launch.
  - May use `/lib` during staging, or make it available in $LD_LIBRARY_PATH during launch.
  - May use `/include`, `/pkgconfig`, or `/env` during staging.
  - May copy files from `/profile.d` to `/tmp/app/.profile.d` during staging.
  - May use the supplied config object in `config.yml` during the staging process.

Deploy Apps with a Custom Buildpack

Once a custom buildpack has been created and pushed to a public git repository, the git URL can be passed via the cf CLI when pushing an app.

For example, for a buildpack that has been pushed to GitHub:

```bash
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git
```

Alternatively, you can use a private git repository, with https and username/password authentication, as follows:

```bash
$ cf push my-new-app -b https://username:password@github.com/johndoe/my-buildpack.git
```

By default, PAS uses the default branch of the buildpack's git repository. You can specify a different branch using the git url as shown in the following example:

```bash
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git#my-branch-name
```

Additionally, you can use tags in a git repository, as follows:

```bash
$ cf push my-new-app -b https://github.com/johndoe/my-buildpack.git#v1.4.2
```

The app will then be deployed to PAS, and the buildpack will be cloned from the repository and applied to the app.

**Note:** If a buildpack is specified using `cf push -b`, the detect step will be skipped and as a result, no buildpack detect scripts will be run.

Disable Custom Buildpacks

Operators can choose to disable custom buildpacks. For more information, see [Disabling Custom Buildpacks](#).

**Note:** A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. To merge upstream patches to your custom buildpack, use the approach that Github recommends for [syncing a fork](#).
Packaging Dependencies for Offline Buildpacks

Page last updated:

This topic describes the dependency storage options available to developers creating offline buildpacks.

About Offline Buildpacks

Online, or uncached, buildpacks require an Internet connection to download dependencies, such as language interpreters and compilers. Alternatively, you can create offline, or cached, buildpacks that are packaged with their dependencies. These offline buildpacks do not connect to the Internet when they are used to deploy Cloud Foundry apps.

Note: Offline buildpacks may contain proprietary dependencies that require distribution licensing or export control measures.

You can find instructions for building the offline packages in the README.md file of each buildpack repository. For example, see the Java buildpack.

Package Dependencies in the Buildpack

The simplest way to package dependencies in a custom buildpack is to keep the dependencies in your buildpack source. However, this is strongly discouraged. Keeping the dependencies in your source consumes unnecessary space.

To avoid keeping the dependencies in source control, load the dependencies into your buildpack and provide a script for the operator to create a zipfile of the buildpack.

For example, the operator might complete the following process:

```bash
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Creates a zipfile using your script
$ .SomeScriptName
    downloading-dependencies....done
    creating zipfile: ZippedBuildPackName.zip

$ # Adds the buildpack zipfile to the Cloud Foundry instance
$ cf create-buildpack SomeBuildPackName ZippedBuildPackName.zip
```

Pros

- Least complicated process for operators
- Least complicated maintenance process for buildpack developers

Cons

- Cloud Foundry admin buildpack uploads are limited to 1 GB, so the dependencies might not fit
- Security and functional patches to dependencies require updating the buildpack

Package Selected Dependencies in the Buildpack

This is a variant of the package dependencies in the buildpack method described above. In this variation, the administrator edits a configuration file such as dependencies.yml to include a limited subset of the buildpack dependencies, then packages and uploads the buildpack.

Note: This approach is strongly discouraged. Please see the Cons section below for more information.

The administrator completes the following steps:
Pros

- Possible to avoid the Cloud Foundry admin buildpack upload size limit in one of two ways:
  - If the administrator chooses a limited subset of dependencies
  - If the administrator maintains different packages for different dependency sets

Cons

- More complex for buildpack maintainers
- Security updates to dependencies require updating the buildpack
- Proliferation of buildpacks that require maintenance:
  - For each configuration, there is an update required for each security patch
  - Culling orphan configurations may be difficult or impossible
  - Administrators need to track configurations and merge them with updates to the buildpack
  - May result in with a different config for each app

Rely on a Local Mirror

In this method, the administrator provides a compatible file store of dependencies. When running the buildpack, the administrator specifies the location of the file store. The buildpack should handle missing dependencies gracefully.

The administrator completes the following process:

Pros
• Avoids the Cloud Foundry admin buildpack upload size limit
• Leaves the administrator completely in control of providing dependencies
• Security and functional patches for dependencies can be maintained separately on the mirror given the following conditions:
  • The buildpack is designed to use newer semantically versioned dependencies
  • Buildpack behavior does not change with the newer functional changes

Cons

• The administrator needs to set up and maintain a mirror
• The additional config option presents a maintenance burden
Merging from Upstream Buildpacks

This topic describes how to maintain your forked buildpack by merging it with the upstream buildpack. This allows you to keep your fork updated with changes from the original buildpack, providing patches, updates, and new features.

The following procedure assumes that you are maintaining a custom buildpack that was forked from a Cloud Foundry system buildpack. However, you can use the same procedure to update a buildpack forked from any upstream buildpack.

To sync your forked buildpack with an upstream Cloud Foundry buildpack:

1. Navigate to your forked repository on GitHub and click Compare in the upper right to display the Comparing changes page. This page shows the unmerged commits between your forked buildpack and the upstream buildpack.

2. Inspect the unmerged commits and confirm that you want to merge them all.

3. In a terminal window, navigate to the forked repository and set the upstream remote as the Cloud Foundry buildpack repository.

   ```bash
   cd ~/workspace/ruby-buildpack
   git remote add upstream git@github.com:cloudfoundry/ruby-buildpack.git
   ```

4. Pull down the remote upstream changes.

   ```bash
   git fetch upstream
   ```

5. Merge the upstream changes into the intended branch. You may need to resolve merge conflicts. This example shows merging the master branch of the upstream buildpack into the master branch of the forked buildpack.

   ```bash
   git checkout master
   git merge upstream/master
   ```

   **Note:** When merging upstream buildpacks, do not use `git rebase`. This approach is not sustainable because you confront the same merge conflicts repeatedly.

6. Run the buildpack test suite to ensure that the upstream changes do not break anything.

   ```bash
   $ BUNDLE_GEMFILE=cf.Gemfile buildpack-build
   ```

7. Push the updated branch.

   ```bash
   git push
   ```

Your forked buildpack is now synced with the upstream Cloud Foundry buildpack.

For more information about syncing forks, see the GitHub topic [Syncing a Fork](https://github.com/).
Upgrading Dependency Versions

Page last updated:

This topic describes how to upgrade a dependency version in a custom buildpack. These procedures enable Cloud Foundry (CF) operators to maintain custom buildpacks that contain dependencies outside of the dependencies in the CF system buildpacks.

Cloud Foundry Buildpacks Team Process

**Note:** The procedures in this topic refer to the tools used by the CF buildpacks team, but they do not require the specific tools listed below. You can use any continuous integration (CI) system and workflow management tool to update dependencies in custom buildpacks.

The CF buildpacks team uses the following tools to update dependencies:

- A [Concourse](#) deployment of the [buildpacks-ci](#) pipelines
- The [public-buildpacks-ci-robots](#) GitHub repository
- [Pivotal Tracker](#) for workflow management

When the [New Releases](#) job in the [notifications pipeline](#) detects a new version of a tracked dependency in a buildpack, it creates a [Tracker](#) story about building and including the new version of the dependency in the buildpack manifests. It also posts a message as the [dependency-notifier](#) to the [#buildpacks](#) channel in the Cloud Foundry Slack channel.

Build the Binaries

For all dependencies, you must build the binary from source or acquire the binary as a tarball from a trusted source. For most dependencies, the CF buildpacks team builds the binaries from source.

**Note:** The steps below assume you are using a Concourse deployment of the [buildpacks-ci](#) pipelines and Pivotal Tracker.

To build the binary for a dependency, perform the following steps:

1. Navigate to the [public-buildpacks-ci-robots](#) directory and verify no uncommitted changes exist.

   ```
   $ cd ~/workspace/public-buildpacks-ci-robots
   $ git status
   ```

2. Run the `git pull` command in the directory to ensure it contains most recent version of the contents.

   ```
   $ git pull -r
   ```

3. Navigate to the [binary-builds](#) directory.

   ```
   $ cd binary-builds
   ```

4. Locate the YAML file for the buildpack you want to build a binary for. The directory contains YAML files for all the packages and dependencies tracked by the CF buildpacks team. Each YAML file correlates to the build queue for one dependency or package, and uses the naming format `DEPENDENCY-NAME.yml`. For example, the YAML file tracking the build queue for Ruby is named `ruby-builds.yml` and contains the following contents:

   ```yaml
   --
   ruby: []
   ```

5. Different buildpacks use different signatures for verification. Determine which signature your buildpack requires by consulting the list in the [buildpacks](#) section of this topic and run follow the instructions to locate the SHA256, MD5, or GPG signature for the binary:

   - For the SHA256 of a file, run `shasum -a 256 FILE-NAME`.
   - For the MD5 of a file, run `md5 FILE-NAME`.
   - For the GPG signature (for Nginx), see the [Nginx Downloads](#) page.

6. Add the version and verification for the new binary to the YAML file as attributes of an element under the dependency name. For example, to build
the Ruby 2.3.0 binary verified with SHA256, add the following to the YAML file:

```
---
ruby:
  - version: 2.3.0
    sha256: ba5ba60e5f1aa21b4ebf9c9f359d7b57286c546aac4b5a28c7f14594e7c507
```

**Note:** Do not preface the version number with the name of the binary or language. For example, specify \(2.3.0\) for `version` instead of `ruby-2.3.0`.

You can enqueue builds for multiple versions at the same time. For example, to build both the Ruby 2.3.0 binary and the Ruby 2.3.1 binary, add the following to the YAML file:

```
---
ruby:
  - version: 2.3.0
    sha256: ba5ba60e5f1aa21b4ebf9c9f359d7b57286c546aac4b5a28c7f14594e7c507
  - version: 2.3.1
    sha256: b87c738cb2032bf4920fef8e3864dc58eae9d8d579d7ac
```

7. Use the `git add` command to stage your changes:

```
$ git add .
```

8. Use the `git commit -m "YOUR-COMMIT-MESSAGE [STORY-NUMBER]"` command to commit your changes using the Tracker story number. Replace `YOUR-COMMIT-MESSAGE` with your commit message, and `STORY-NUMBER` with the number of your Tracker story.

```
$ git commit -m "make that change [#1234567890]"
```

9. Run `git push` to push your changes to the remote origin.

```
$ git push
```

10. Pushing your changes triggers the binary building process, which you can monitor at the binary-builder pipeline of your own buildpacks-ci Concourse deployment. When the build completes, it adds a link to the Concourse build run to the Tracker story specified in the commit message for the new release.

**Note:** Binary builds are executed by the Cloud Foundry Binary Builder and the binary-builder-pipeline.

---

**Update Buildpack Manifests**

After you build the binary for a dependency that you can access and download from a URL, follow the instructions below to add the dependency version to the buildpack manifest.

**Note:** The steps below assume you are using a Concourse deployment of the buildpacks-ci pipelines and Pivotal Tracker.

1. Navigate to the directory of the buildpack for which you want to update dependencies and run `git checkout develop` to check out the develop branch.

```
$ cd ~/workspace/ruby-buildpack
$ git checkout develop
```

2. Edit the `manifest.yml` file for the buildpack to add or remove dependencies.

```
dependencies:
  - name: ruby
    version: 2.3.0
    sha256: 535342030d1af914c02f149f24b90e5f12
    url: https://pivotal-buildpacks.s3.amazonaws.com/concourse-binaries/ruby/ruby-2.3.0-linux-x64.tgz
    cf_stacks:
      - cflinuxfs2
```

* Follow the current structure of the manifest. For example, if the manifest includes the two most recent patch versions for each minor version of
the language, you should also include the two most recent patch versions for each minor version of the language, such as both ruby-2.1.9 and ruby-2.1.8.

- Copy the `uri` and the `md5` from the `build-BINARY-NAME` job that ran in the Concourse binary-builder pipeline and add them to the manifest.

Note: In the PHP buildpack, you may see a `modules` line for each PHP dependency in the manifest. Do not include this `modules` line in your new PHP dependency entry. The `modules` line will be added to the manifest by the `ensure-manifest-has-modules` Concourse job in the `php-buildpack` when you commit and push your changes. You can see this in the output logs of the `build-out` task.

3. Replace any other mentions of the old version number in the buildpack repository with the new version number. The CF buildpack team uses `Ag` for text searching.

```
$ ag OLD-VERSION
```

4. Run the following command to package and upload the buildpack, set up the org and space for tests in the specified CF deployment, and run the CF buildpack tests.

```
$ BUNDLE_GEMFILE=cf.Gemfile buildpack-build
```

If the command fails, you may need to fix or change the tests, fixtures, or other parts of the buildpack.

5. Once the test suite completely passes, use git commands to stage, commit, and push your changes:

```
$ git add .
$ git commit -m "YOUR-MESSAGE[#TRACKER-STORY-ID]"
$ git push
```

6. Monitor the `LANGUAGE-buildpack` pipeline in Concourse. Once the test suite builds, the `specs-lts-develop` and `specs-edge-develop` job, pass for the buildpack, you can deliver the Tracker story for the new Dependency release. Copy and paste links for the successful test suite builds into the Tracker story.

**Buildpacks**

The following list contains information about the buildpacks maintained by the CF buildpacks team.

**Go Buildpack**

Go:

- **Built from:** A tarred binary, `GO-VERSION.linux-amd64.tar.gz`, provided by [Google on the Go Downloads page](https://golang.org/doc/go1.6.2.html)
- **Verified with:** The MD5 of the tarred binary
- **Example:** [Using the Google Tarred Binary for Go 1.6.2](https://golang.org/doc/go1.6.2.html)

Godep:

- **Built from:** A source code `.tar.gz` file from the [Godep GitHub releases page](https://github.com/golang/deps/releases)
- **Verified with:** The SHA256 of the source
- **Example:** [Automated enqueuing of binary build for Godep T2](https://github.com/golang/deps/releases)

> Note: The `buildpacks-ci binary-builder` pipeline automates the process of detecting, uploading, and updating Godep in the manifest.

**Node.js Buildpack**

Node:

- **Verified with:** The SHA256 of the `node-VERSION.tar.gz` file listed on [Node.js](https://nodejs.org/dist/v4.4.6/SHASUMS256.txt) for Node version 4.4.6. The CF buildpacks team verifies with the SHA256 for `node-v4.4.6.tar.gz` on its [SHASUMS256](https://github.com/nodejs/ dist/v4.4.6/SHASUMS256.txt) page.
- **Example:** [Enqueuing binary builds for Node 4.4.5 and 6.2.0](https://github.com/nodejs/dist/v4.4.6/SHASUMS256.txt)
Python Buildpack

Python:

- **Verified with**: The MD5 of the Gzipped source tarball, listed on [https://www.python.org/downloads/release/python-VERSION/](https://www.python.org/downloads/release/python-VERSION/), where `VERSION` has no periods. For example, for Python version 2.7.12, use the MD5 for the Gzipped source tarball on its downloads page.

- **Example**: Enqueuing binary build for Python 2.7.12

Java Buildpack

OpenJDK:

- **Built from**: The tarred [OpenJDK](https://www.oracle.com/java/technologies/jdk-downloads.html) managed by the CF Java Buildpack team

- **Verified with**: The MD5 of the tarred OpenJDK files

Ruby Buildpack

JRuby:

- **Verified with**: The MD5 of the Source `.tar.gz` file from the [JRuby Downloads](https://github.com/jruby/jruby/downloads) page

- **Example**: Enqueuing binary build for JRuby 9.1.2.0

Ruby:

- **Verified with**: The SHA256 of the source from the [Ruby Downloads](https://github.com/ruby/ruby/downloads) page

- **Example**: Enqueuing binary builds for Ruby 2.2.5 and 2.3.1

Bundler:

- **Verified with**: The SHA256 of the `.gem` file from [Rubygems](https://rubygems.org)

- **Example**: Enqueuing binary build for Bundler 1.12.5

PHP Buildpack

PHP:

- **Verified with**: The SHA256 of the `.tar.gz` file from the [PHP Downloads](https://github.com/pharFileManager/PHAR-Manager/downloads) page.

To enqueue builds for PHP, you need to edit a file in the `public-buildpacks-ci-robots` repository. For PHP5 versions, the CF buildpacks team enqueues builds in the `binary-builds/php-builds.yml` file. For PHP7 versions, the CF buildpacks team enqueues builds in the `binary-builds/php7-builds.yml` file.

- **Example**: Enqueuing binary builds for PHP 7.2.5 and 7.0.30

Nginx:

- **Verified with**: The `gpg-rsa-key-id` and `gpg-signature` of the version. The `gpg-rsa-key-id` is the same for each version/build, but the `gpg-signature` will be different. This information is located on the [Nginx Downloads](https://nginx.org/download) page.

- **Example**: Enqueuing binary build for Nginx 1.11.0

HTTPD:

- **Verified with**: The MD5 of the `.tar.bz2` file from the [HTTPD Downloads](https://httpd.apache.org/download.html) page

- **Example**: Enqueuing binary build for HTTPD 2.4.20

Composer:

- **Verified with**: The SHA256 of the `composer.phar` file from the [Composer Downloads](https://getcomposer.org/downloads) page

For Composer, there is no build process as the `composer.phar` file is the binary. In the manual process, connect to the appropriate S3 bucket using the correct AWS credentials. Create a new directory with the name of the composer version, for example `composer.phar`, and put the appropriate `composer.phar` file into that directory. For Composer `v1.0.2`, connect and create the `/php/binaries/trusty/composer/1.0.2` directory. Then place the `composer.phar` file into that directory so the binary is available at `/php/binaries/trusty/composer/1.0.2/composer.phar`.
Staticfile Buildpack

Nginx:

- **Verified with**: The `gpg-rsa-key-id` and `gpg-signature` of the version. The `gpg-rsa-key-id` is the same for each version/build, but the `gpg-signature` will be different. This information is located on the [Nginx Downloads](#) page.
- **Example**: [Enqueuing binary build for Nginx 1.11.0](#)

Binary Buildpack

The Binary buildpack has no dependencies.
Using CI for Buildpacks

The Cloud Foundry (CF) Buildpacks team and other CF buildpack development teams use Concourse continuous integration (Concourse CI) pipelines to integrate new buildpack releases. This topic provides links to information that describes how to release new versions of Cloud Foundry buildpacks using Concourse CI, and how to update Ruby gems used for CF buildpack development.

Each of the following are applicable to all supported buildpack languages and frameworks:

- Releasing a New Buildpack Version
- Updating Buildpack-Related Gems
Releasing a New Buildpack Version

To release a new buildpack version, perform the following:

1. Ensure you have downloaded the buildpacks-ci repository:
   
   ```
   $ git clone https://github.com/cloudfoundry/buildpacks-ci.git
   ```

2. From the buildpack directory, check out the develop branch of the buildpack:
   
   ```
   $ cd /system/path/to/buildpack
   $ git checkout develop
   ```

3. Ensure you have the most current version of the repository:
   
   ```
   $ git pull -r
   ```

4. Run `bump` to update the version in the buildpack repository:
   
   ```
   $ /system/path/to/buildpacks-ci/scripts/bump
   ```

5. Modify the CHANGELOG file manually to condense recent commits into relevant changes. For more information, see Modify Changelogs.

6. Add and commit your changes:
   
   ```
   $ git add VERSION CHANGELOG
   $ git commit -m "Bump version to $(cat VERSION) [{insert story #}]"
   ```

7. Push your changes to the develop branch:
   
   ```
   $ git push origin develop
   ```

Concourse Buildpack Workflow

If buildpacks-ci is not deployed to Concourse, manually add a Git tag to the buildpack and mark the tag as a release on GitHub.

If buildpacks-ci is deployed to Concourse, the buildpack update passes through the following life cycle:

1. Concourse triggers the buildpack-to-master job in the pipeline for the updated buildpack. This job merges develop onto the master branch of the buildpack.

2. The detect-new-buildpack-and-upload-artifacts job triggers in the pipeline for the updated buildpack. This job creates a cached and uncached buildpack and uploads them to an AWS S3 bucket.

3. The specs-lts-master and specs-edge-master jobs trigger and run the buildpack test suite and the buildpack-specific tests of the Buildpack Runtime Acceptance Tests (BRATS).

4. If you are using Pivotal Tracker, paste the links for the specs-edge-master and specs-lts-master builds in the related buildpack release story and deliver that story.
5. Your project manager can manually trigger the `buildpack-to-github` job on Concourse as part of the acceptance process. This releases the buildpack to GitHub.

6. After the buildpack has been released to GitHub, the `cf-release` pipeline is triggered using the manual trigger of the `remote-bosh-lite` job on that pipeline. If the new buildpack has been released to GitHub, the CF that is deployed for testing in the `cf-release` pipeline is tested against that new buildpack.

7. After the `cats` job has successfully completed, your project manager can ship the new buildpacks to the `cf-release` repository and create the new buildpack BOSH release by manually triggering the `ship-it` job.

**Note:** If errors occur during this workflow, you may need to remove unwanted tags. For more information, see **Handle Unwanted Tags**.

### Modify Changelogs

The **Ruby Buildpack changelog** shows an example layout and content of a changelog. In general, changelogs follow these conventions:

- Reference public tracker stories whenever possible.
- Exclude unnecessary files
- Combine and condense commit statements into individual stories containing valuable changes.

### Handle Unwanted Tags

If you encounter problems with the commit that contains the new version, change the target of the release tag by performing the following:

1. Ensure the repository is in a valid state and is building successfully.
2. Remove the tag from your local repository and from GitHub.
3. Start a build. The pipeline build script should re-tag the build if it is successful.
Updating Buildpack-Related Gems

Page last updated:

This topic describes how to update buildpack-packager and machete, used for CF system buildpack development.

buildpack-packager packages buildpacks and machete provides an integration test framework.

The CF Buildpacks team uses the gems-and-extensions pipeline to:

1. Run the integration tests for buildpack-packager and machete
2. Update the gems in the buildpacks managed by the team

Running the Update Process

Note: The steps below assume you are using a Concourse deployment of the buildpacks-ci pipelines

At the end of the process, there will be a new GitHub release and updates will be applied to the buildpacks.

To update the version of either gem in a buildpack:

1. Confirm that the test job <gемname>-specs for the gem to be updated successfully ran on the commit you plan to update.
2. Manually trigger the <gемname>-tag job to update (“bump”) the version of the gem.
3. The <gемname>-release job will trigger. This will create a new GitHub release of the gem.
4. Each of the buildpack pipelines (e.g. the go-buildpack pipeline) has a job which watches for new releases of the gem. When a new release is detected, the buildpack's cf.Gemfile is updated to that release version.
5. The commit made to the buildpack’s cf.Gemfile triggers the full integration test suite for that buildpack.

Note: The final step will trigger all buildpack test suites simultaneously, causing contention for available shared BOSH-lite test environments.
The documentation in this section is intended for developers and operators interested in creating Managed Services for Cloud Foundry. Managed Services are defined as having been integrated with Cloud Foundry via APIs, and enable end users to provision reserved resources and credentials on demand. For documentation targeted at end users, such as how to provision services and integrate them with applications, see Services Overview.

To develop Managed Services for Cloud Foundry, you’ll need a Cloud Foundry instance to test your service broker with as you are developing it. You must have admin access to your CF instance to manage service brokers and the services marketplace catalog. For local development, we recommend using the BOSH guide to deploy your own local instance of Cloud Foundry using VirtualBox.

Table of Contents
- Overview
- Service Broker API
  - Open Service Broker API
  - Platform Profiles
  - Catalog Metadata
  - Volume Services
  - Release Notes
- Managing Service Brokers
- Access Control
- Dashboard Single Sign-On
- Example Service Brokers
- Binding Credentials
- Enabling Service Instance Sharing
- Application Log Streaming
- Route Services
- Supporting Multiple Cloud Foundry Instances
Overview

Page last updated:

Architecture & Terminology

Services are integrated with Cloud Foundry by implementing a documented API for which the cloud controller is the client; we call this the Service Broker API. This should not be confused with the cloud controller API, often used to refer to the version of Cloud Foundry itself; when one refers to “Cloud Foundry v2” they are referring to the version of the cloud controller API. The services API is versioned independently of the cloud controller API.

Service Broker is the term we use to refer to a component of the service which implements the service broker API. This component was formerly referred to as a Service Gateway, however as traffic between applications and services does not flow through the broker we found the term gateway caused confusion. Although gateway still appears in old code, we use the term broker in conversation, in new code, and in documentation.

Service brokers advertise a catalog of service offerings and service plans, as well as interpreting calls for provision (create), bind, unbind, and deprovision (delete). What a broker does with each call can vary between services; in general, ‘provision’ reserves resources on a service and ‘bind’ delivers information to an application necessary for accessing the resource. We call the reserved resource a Service Instance. What a service instance represents can vary by service; it could be a single database on a multi-tenant server, a dedicated cluster, or even just an account on a web application.

Implementation & Deployment

How a service is implemented is up to the service provider/developer. Cloud Foundry only requires that the service provider implement the service broker API. A broker can be implemented as a separate application, or by adding the required http endpoints to an existing service.

Because Cloud Foundry only requires that a service implements the broker API in order to be available to Cloud Foundry end users, many deployment models are possible. The following are examples of valid deployment models.

- Entire service packaged and deployed by BOSH alongside Cloud Foundry
- Broker packaged and deployed by BOSH alongside Cloud Foundry, rest of the service deployed and maintained by other means
- Broker (and optionally service) pushed as an application to Cloud Foundry user space
- Entire service, including broker, deployed and maintained outside of Cloud Foundry by other means
Managing Service Brokers

Page last updated:

This page assumes you are using cf CLI v6.16 or later.

In order to run many of the commands below, you must be authenticated with Cloud Foundry as an admin user or as a space developer.

Quick Start

Given a service broker that has implemented the Service Broker API, two steps are required to make its services available to end users in all orgs or a limited number of orgs by service plan.

1. Register a Broker
2. Make Plans Public

Register a Broker

Registering a broker causes Cloud Controller to fetch and validate the catalog from your broker, and save the catalog to the Cloud Controller database. The basic auth username and password which are provided when adding a broker are encrypted in Cloud Controller database, and used by the Cloud Controller to authenticate with the broker when making all API calls. Your service broker should validate the username and password sent in every request; otherwise, anyone could curl your broker to delete service instances. When the broker is registered with an URL having the scheme https, Cloud Controller will make all calls to the broker over HTTPS.

As of cf-release 229, CC API 2.47.0, Cloud Foundry supports two types of brokers: standard brokers and space-scoped brokers. A list of their differences follows:

**Standard Brokers**

- Publish service plans to specific orgs or all orgs in the deployment. Can also keep plans unavailable, or private.
- Created by admins, with the command `cf create-service-broker`:
  
  ```
  $ cf create-service-broker mybrokername someuser somethingsecure https://mybroker.example.com
  ```
- Managed by admins
- Service plans are created private. Before anyone can use them, an admin must explicitly make them available within an org or across all orgs.

**Space-Scoped Brokers**

- Publish service plans only to users within the space they are created. Plans are unavailable outside of this space.
- Created by space developers using the command `cf create-service-broker` with the `--space-scoped` flag:
  
  ```
  $ cf create-service-broker mybrokername someuser somethingsecure https://mybroker.example.com --space-scoped
  ```
- Managed by space developers
- Newly-created plans automatically publish to all users in their space.

Make Plans Public

After an admin creates a new service plan from a standard broker, no one can use it until the admin explicitly makes it available to users within a specific org or all orgs in the deployment.

See the Access Control topic for how to make standard broker service plans available to users.
Multiple Brokers, Services, Plans

Many service brokers may be added to a Cloud Foundry instance, each offering many services and plans. The following constraints should be kept in mind:

- It is not possible to have multiple brokers with the same name
- The service ID and plan IDs of each service advertised by the broker must be unique across Cloud Foundry. GUIDs are recommended for these fields.

Note: If your deployment uses Cloud Foundry API (CAPI) v1.71 or later, you can add multiple brokers with the same URL. In this case, the brokers must have different names. CAPI v1.70 and earlier do not support this feature.

See Possible Errors below for error messages and what to do when you see them.

List Service Brokers

$ cf service-brokers
Getting service brokers as admin...
OK

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-service-broker</td>
<td><a href="https://mybroker.example.com">https://mybroker.example.com</a></td>
</tr>
</tbody>
</table>

Update a Broker

Updating a broker is how to ingest changes a broker author has made into Cloud Foundry. Similar to adding a broker, update causes Cloud Controller to fetch the catalog from a broker, validate it, and update the Cloud Controller database with any changes found in the catalog.

Update also provides a means to change the basic auth credentials cloud controller uses to authenticate with a broker, as well as the base URL of the broker’s API endpoints.

$ cf update-service-broker mybrokername someuser somethingsecure https://mybroker.example.com

Rename a Broker

A service broker can be renamed with the rename-service-broker command. This name is used only by the Cloud Foundry operator to identify brokers, and has no relation to configuration of the broker itself.

$ cf rename-service-broker mybrokername mynewbrokername

Remove a Broker

Removing a service broker will remove all services and plans in the broker's catalog from the Cloud Foundry Marketplace.

$ cf delete-service-broker mybrokername

Note: Attempting to remove a service broker will fail if there are service instances for any service plan in its catalog. When planning to shut down or delete a broker, make sure to remove all service instances first. Failure to do so will leave orphaned service instances in the Cloud Foundry database. If a service broker has been shut down without first deleting service instances, you can remove the instances with the CLI; see Purge a Service.

Purge a Service

If a service broker has been shut down or removed without first deleting service instances from Cloud Foundry, you will be unable to remove the service...
broker or its services and plans from the Marketplace. In development environments, broker authors often destroy their broker deployments and need a way to clean up the Cloud Controller database.

The following command will delete a service offering, all of its plans, as well as all associated service instances and bindings from the Cloud Controller database, without making any API calls to a service broker. Once all services for a broker have been purged, the broker can be removed normally.

```bash
cf purge-service-offering service-test
```

Warning: This operation assumes that the service broker responsible for this service offering is no longer available, and all service instances have been deleted, leaving orphan records in Cloud Foundry’s database. All knowledge of the service will be removed from Cloud Foundry, including service instances and service bindings. No attempt will be made to contact the service broker; running this command without destroying the service broker will cause orphan service instances. After running this command you may want to run either `delete-service-auth-token` or `delete-service-broker` to complete the cleanup.

Really purge service offering service-test from Cloud Foundry? y

OK

**Note:** To purge services from v1 brokers, you must append your `cf purge-service-offering` command with -p PROVIDER.

---

**Purge a Service Instance**

The following command will delete a single service instance, its service bindings and its service keys from the Cloud Controller database, without making any API calls to a service broker. This can be helpful in instances a Service Broker is not conforming to the Service Broker API and not returning a 200 or 410 to requests to delete the service instance.

```bash
cf purge-service-instance mysql-dev
```

WARNING: This operation assumes that the service broker responsible for this service instance is no longer available or is not responding with a 200 or 410, and the service instance has been deleted, leaving orphan records in Cloud Foundry’s database. All knowledge of the service instance will be removed from Cloud Foundry, including service bindings and service keys.

Really purge service instance mysql-dev from Cloud Foundry? y

Purging service mysql-dev...

OK


**Note:** When multiple brokers provide two or more service instances with the same name, you must specify the broker by including the -b BROKER flag in the `cf purge-service-offering` command.

---

**Catalog Validation Behaviors**

When Cloud Foundry fetches a catalog from a broker, it will compare the broker’s id for services and plans with the `unique_id` values for services and plans in the Cloud Controller database.

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>The catalog fails to load or validate.</td>
<td>Cloud Foundry will return a meaningful error that the broker could not be reached or the catalog was not valid.</td>
</tr>
<tr>
<td>A service or plan in the broker catalog has an ID that is not present among the <code>unique_id</code> values in the marketplace database.</td>
<td>A new record must be added to the marketplace database.</td>
</tr>
<tr>
<td>A service or plan in the marketplace database are found with a <code>unique_id</code> that matches the broker catalog’s ID.</td>
<td>The marketplace must update the records to match the broker’s catalog.</td>
</tr>
<tr>
<td>The database has plans that are not found in the broker catalog, and there are no associated service instances.</td>
<td>The marketplace must remove these plans from the database, and then delete services that do not have associated plans from the database.</td>
</tr>
<tr>
<td>The database has plans that are not found in the broker catalog, but there are provisioned instances.</td>
<td>The marketplace must mark the plan inactive and no longer display it or allow it to be provisioned.</td>
</tr>
</tbody>
</table>
Possible Errors

If incorrect basic auth credentials are provided:

Server error, status code: 500, error code: 10001, message: Authentication failed for the service broker API.
Double-check that the username and password are correct:
https://github-broker.a1-app.example.com/v2/catalog

If you receive the following errors, check your broker logs. You may have an internal error.

Server error, status code: 500, error code: 10001, message: The service broker response was not understood
Server error, status code: 500, error code: 10001, message: The service broker API returned an error from
https://github-broker.a1-app.example.com/v2/catalog: 404 Not Found
Server error, status code: 500, error code: 10001, message: The service broker API returned an error from
https://github-broker.primo.example.com/v2/catalog: 500 Internal Server Error

If your broker's catalog of services and plans violates validation of presence, uniqueness, and type, you will receive meaningful errors.

Server error, status code: 502, error code: 270012, message: Service broker catalog is invalid:
Service service-name-1
service id must be unique
service description is required
service "bindable" field must be a boolean, but has value "true"
Plan plan-name-1
plan metadata must be a hash, but has value [{"bullets"="["bullet1","bullet2"]"}]
Managing Access to Service Plans

Page last updated:

All new service plans from standard brokers are private by default. This means that when adding a new broker, or when adding a new plan to an existing broker’s catalog, service plans won’t immediately be available to end users. This lets an admin control which service plans are available to end users, and manage limited service availability.

Space-scoped brokers are registered to a specific space, and all users within that space can automatically access the broker’s service plans. With space-scoped brokers, service visibility is not managed separately.

Prerequisites

- CLI v6.4.0
- Cloud Controller API v2.9.0 (cf-release v179)
- Admin user access; the following commands can be run only by an admin user

Display Access to Service Plans

The `cf service-access` CLI command enables an admin to see the current access control setting for every service plan in the marketplace, across all service brokers.

```bash
$ cf service-access
getting service access as admin...
brokers: elasticsearch-broker
  service plan access orgs
  elasticsearch standard limited
brokers: p-mysql
  service plan access orgs
  p-mysql 100mb-dev all
```

The `access` column shows values `all`, `limited`, or `none`, defined as follows:

- **all**: The service plan is available to all users, or public.
- **none**: No one can use the service plan; it is private.
- **limited**: The plan is available only to users within the orgs listed.

The `-b`, `-e`, and `-o` flags let you filter by broker, service, and org.

```bash
$ cf help service-access
NAME:
  service-access - List service access settings

USAGE:
  cf service-access [-b BROKER] [-e SERVICE] [-o ORG]

OPTIONS:
  -b  access for plans of a particular broker
  -e  access for plans of a particular service offering
  -o  plans accessible by a particular org
```

Enable Access to Service Plans

Admins use the `cf enable-service-access` command to give users access to service plans. The command grants access at the org level or across all orgs.

When an org has access to a plan, its users see the plan in the services marketplace (`cf marketplace`) and its Space Developer users can provision instances of the plan in their spaces.
Enable Access to a Subset of Users

The `-p` and `-o` flags to `cf enable-service-access` let the admin limit user access to specific service plans or orgs as follows:

- `-p PLAN` grants all users access to one service plan (access: all)
- `-o ORG` grants users in a specified org access to all plans (access: limited)
- `-p PLAN -o ORG` grants users in one org access to one plan (access: limited)

For example, the following command grants the org dev-user-org access to the p-mysql service.

```
$ cf enable-service-access p-mysql -o dev-user-org
```

Enabling access to all plans of service p-mysql for the org dev-user-org as admin...

OK

```
$ cf service-access
get service access as admin...
broker: p-mysql
  service plan  access  orgs
p-mysql  dev-user-org
```

Run `cf help enable-service-access` to review these options from the command line.

Running `cf enable-service-access SERVICE-NAME` without any flags lets all users access every plan carried by the service. For example, the following command grants all-user access to all p-mysql service plans:

```
$ cf enable-service-access p-mysql
Enabling access to all plans of service p-mysql for all orgs as admin...
OK

$ cf service-access
get service access as admin...
broker: p-mysql
  service plan  access  orgs
p-mysql  100mb-dev  all
```

**Note:** When multiple brokers provide two or more service instances with the same name, you must specify the broker by including the `-b BROKER` flag in the `cf enable-service-access` command.

Disable Access to Service Plans

Admins use the `cf disable-service-access` command to disable user access to service plans. The command denies access at the org level or across all orgs.

Disable Access to All Plans for All Users

Running `cf disable-service-access SERVICE-NAME` without any flags disables all user access to all plans carried by the service. For example, the following command denies any user access to all p-mysql service plans:

```
$ cf disable-service-access p-mysql
Disabling access to all plans of service p-mysql for all orgs as admin...
OK

$ cf service-access
get service access as admin...
broker: p-mysql
  service plan  access  orgs
p-mysql  100mb-dev  none
```

Disable Access for Specific Orgs or Plans

The `-p` and `-o` flags to `cf disable-service-access` let the admin deny access to specific service plans or orgs as follows:
-p PLAN — disables user access to one service plan
-o ORG — disables access to all plans for users in a specified org
-p PLAN -o ORG — prevents users in one org from accessing one plan

Run `cf help disable-service-access` to review these options from the command line.

Note: When multiple brokers provide two or more service instances with the same name, you must specify the broker by including the `--broker` flag in the `cf disable-service-access` command.

Limitations
- You cannot disable access to a service plan for an org if the plan is currently available to all orgs. You must first disable access for all orgs; then you can enable access for a particular org.
Dashboard Single Sign-On

Page last updated:

Introduction

Single Sign-On (SSO) enables Pivotal Cloud Foundry (PCF) users to authenticate with third-party service dashboards using their PCF credentials. Service dashboards are web interfaces which enable users to interact with some or all of the features the service offers. SSO provides a streamlined experience for users, limiting repeated logins and multiple accounts across their managed services. The user's credentials are never directly transmitted to the service since the OAuth protocol handles authentication.

Service Broker Responsibilities

Registering the Dashboard Client

1. A service broker must include the `dashboard_client` field in the JSON response from its catalog endpoint for each service implementing this feature. A valid response would appear as follows:

```json
{
  "services": [
    {
      "id": "44b26033-1f54-4087-b7bc-da9652c2a539",
      "dashboard_client": {
        "id": "p-mysql-client",
        "secret": "p-mysql-secret",
        "redirect_uri": "http://p-mysql.example.com"
      }
    }
  ]
}
```

The `dashboard_client` field is a hash containing three fields:

- `id` is the unique identifier for the OAuth client that will be created for your service dashboard on the token server (UAA), and will be used by your dashboard to authenticate with the token server (UAA). If the client id is already taken, PCF will return an error when registering or updating the broker.
- `secret` is the shared secret your dashboard will use to authenticate with the token server (UAA).
- `redirect_uri` is used by the token server as an additional security precaution. UAA will not provide a token if the callback URL declared by the service dashboard doesn't match the domain name in `redirect_uri`. The token server matches on the domain name, so any paths will also match; e.g., a service dashboard requesting a token and declaring a callback URL of `http://p-mysql.example.com/manage/auth` would be approved if `redirect_uri` for its client is `http://p-mysql.example.com/`.

2. When a service broker which advertises the `dashboard_client` property for any of its services is added or updated, Cloud Controller will create or update UAA clients as necessary. This client will be used by the service dashboard to authenticate users.

Dashboard URL

A service broker should return a URL for the `dashboard_url` field in response to a provision request. Cloud Controller clients should expose this URL to users. `dashboard_url` can be found in the response from Cloud Controller to create a service instance, enumerate service instances, space summary, and other endpoints.

Users can then navigate to the service dashboard at the URL provided by `dashboard_url`, initiating the OAuth login flow.

Service Dashboard Responsibilities

© Copyright Pivotal Software Inc, 2013-2019

1495
OAuth Flow
When a user navigates to the URL from dashboard_url , the service dashboard should initiate the OAuth login flow. A summary of the flow can be found in
section 1.2 of the OAuth RFC . OAuth expects the presence of an Authorization Endpoint  and a Token Endpoint . In PCF, these endpoints are
provided by the UAA. Clients can discover the location of UAA from Cloud Controller’s info endpoint; in the response the location can be found in the
token_endpoint field.
$ curl api.example.com/info
{"name":"vcap","build":"2222","support":"http://support.example.com","version
":2,"description":"Cloud Foundry sponsored by Example Company","authorization_endpoint":
"https://login.example.com","token_endpoint":"https://uaa.example.com",
"allow_debug":true}



To enable service dashboards to support SSO for service instances created from different PCF instances, the /v2/info url is sent to service brokers
in the X-Api-Info-Location header of every API call. A service dashboard should be able to discover this URL from the broker, and enabling the
dashboard to contact the appropriate UAA for a particular service instance.

A service dashboard should implement the OAuth Authorization Code Grant type ( UAA documentation

,

RFC documentation

).

1. When a user visits the service dashboard at the value of dashboard_url , the dashboard should redirect the user’s browser to the Authorization
Endpoint and include its client_id , a redirect_uri (callback URL with domain matching the value of dashboard_client.redirect_uri ), and list of requested
scopes.
Scopes are permissions included in the token a dashboard client will receive from UAA, and which Cloud Controller uses to enforce access. A client
should request the minimum scopes it requires. The minimum scopes required for this workflow are cloud_controller_service_permissions.read and
openid . For an explanation of the scopes available to dashboard clients, see On Scopes.
2. UAA authenticates the user by redirecting the user to the Login Server, where the user then approves or denies the scopes requested by the service
dashboard. The user is presented with human readable descriptions for permissions representing each scope. After authentication, the user’s
browser is redirected back to the Authorization endpoint on UAA with an authentication cookie for the UAA.
3. Assuming the user grants access, UAA redirects the user’s browser back to the value of redirect_uri the dashboard provided in its request to the
Authorization Endpoint. The Location header in the response includes an authorization code.
HTTP/1.1 302 Found
Location: https://p-mysql.example.com/manage/auth?code=F45jH

4. The dashboard UI should then request an access token from the Token Endpoint by including the authorization code received in the previous step.
When making the request the dashboard must authenticate with UAA by passing the client id and secret in a basic auth header. UAA will verify that
the client id matches the client it issued the code to. The dashboard should also include the redirect_uri used to obtain the authorization code for
verification.
5. UAA authenticates the dashboard client, validates the authorization code, and ensures that the redirect URI received matches the URI used to
redirect the client when the authorization code was issues. If valid, UAA responds back with an access token and a refresh token.

Checking User Permissions
UAA is responsible for authenticating a user and providing the service with an access token with the requested permissions. However, after the user has
been logged in, it is the responsibility of the service dashboard to verify that the user making the request to manage an instance currently has access to
that service instance.
The service can accomplish this with a GET to the /v2/service_instances/:guid/permissions endpoint on the Cloud Controller. The request must include a token
for an authenticated user and the service instance guid. The token is the same one obtained from the UAA in response to a request to the Token Endpoint,
described above. .
Example Request:
curl -H 'Content-Type: application/json' \
-H 'Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoid' \
http://api.cloudfoundry.com/v2/service_instances/44b26033-1f54-4087-b7bc-da9652c2a539/permissions

Response:

© Copyright Pivotal Software Inc, 2013-2019

1496

2.5


The response includes the following fields which indicate the various user permissions for the given service instance:

- `manage` - a `true` value indicates that the user has sufficient permissions to make changes to and update the service instance; `false` indicates insufficient permissions.

- `read` - a `true` value indicates that the user has permission to access read-only diagnostic and monitoring information for the given service instance (e.g. permission to view a read-only dashboard); `false` indicates insufficient permissions.

Since administrators may change the permissions of users at any time, the service should check this endpoint whenever a user uses the SSO flow to access the service's UI.

**On Scopes**

Scopes let you specify exactly what type of access you need. Scopes limit access for OAuth tokens. They do not grant any additional permission beyond that which the user already has.

**Minimum Scopes**

The following two scopes are necessary to implement the integration. Most dashboard shouldn't need more permissions than these scopes enabled.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>openid</td>
<td>Allows access to basic data about the user, such as email addresses</td>
</tr>
<tr>
<td>cloud_controller_service_permissions.read</td>
<td>Allows access to the CC endpoint that specifies whether the user can manage a given service instance</td>
</tr>
</tbody>
</table>

**Additional Scopes**

Dashboards with extended capabilities may need to request these additional scopes:

<table>
<thead>
<tr>
<th>Scope</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloud_controller.read</td>
<td>Allows read access to all resources the user is authorized to read</td>
</tr>
<tr>
<td>cloud_controller.write</td>
<td>Allows write access to all resources the user is authorized to update / create / delete</td>
</tr>
</tbody>
</table>

**Reference Implementation**

The MySQL Service Broker is an example of a broker that also implements a SSO dashboard. The login flow is implemented using the OmniAuth library and a custom UAA OmniAuth Strategy. See this OmniAuth wiki page for instructions on how to create your own strategy.

The UAA OmniAuth strategy is used to first get an authorization code, as documented in this section of the UAA documentation. The user is redirected back to the service (as specified by the `callback_path` option or the default `auth/callback` path) with the authorization code. Before the application / action is dispatched, the OmniAuth strategy uses the authorization code to get a token and uses the token to request information from UAA to fill the `omniauth.auth` environment variable. When OmniAuth returns control to the application, the `omniauth.auth` environment variable hash will be filled with the token and user information obtained from UAA as seen in the Auth Controller.

**Restrictions**

- UAA clients are scoped to services. There must be a `dashboard_client` entry for each service that uses SSO integration.

- Each `dashboard_client id` must be unique across the CloudFoundry deployment.

**Resources**

- OAuth
- Example broker with SSO implementation
- Cloud Controller API Docs
- User Account and Authentication (UAA) Service APIs
Example Service Brokers

The following example service broker applications have been developed - these are a great starting point if you are developing your own service broker.

Ruby

- [GitHub Repository service](#) - this is designed to be an easy-to-read example of a service broker, with complete documentation, and comes with a demo app that uses the service. The broker can be deployed as an application to any Cloud Foundry instance or hosted elsewhere. The service broker uses GitHub as the service back end.

- [MySQL database service](#) - this broker and its accompanying MySQL server are designed to be deployed together as a BOSH release. BOSH is used to deploy or upgrade the release, monitors the health of running components, and restarts or recreates unhealthy VMs. The broker code alone can be found [here](#).

Java

- [Spring Cloud - Cloud Foundry Service Broker](#) - This implements the REST contract for service brokers and the artifacts are published to the Spring Maven repository. This greatly simplifies development: include a single dependency in Gradle, implement interfaces, and configure. A sample implementation has been provided for the [MongoDB](#).

- [MySQL Java Broker](#) - a Java port of the Ruby-based MySQL broker above.

Go

- [Asynchronous Service Broker for AWS EC2](#) - This broker implements support for Asynchronous Service Operations, and calls AWS APIs to provision EC2 VMs.
Binding Credentials

Page last updated:

A bindable service returns credentials that an application can consume in response to the `cf bind` API call. Cloud Foundry writes these credentials to the `VCAP_SERVICES` environment variable. In some cases, buildpacks write a subset of these credentials to other environment variables that frameworks might need.

Choose from the following list of credential fields if possible, though you can provide additional fields as needed. Refer to the Using Bound Services section of the Managing Service Instances with the CLI topic for information about how these credentials are consumed.

Note: If you provide a service that supports a connection string, provide the `uri` key for buildpacks and application libraries to use.

<table>
<thead>
<tr>
<th>CREDENTIALS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>Connection string of the form <code>DB-TYPE://USERNAME:PASSWORD@HOSTNAME:PORT/NAME</code>, where <code>DB-TYPE</code> is a type of database such as mysql, postgres, mongodb, or amqp.</td>
</tr>
<tr>
<td>hostname</td>
<td>FQDN of the server host</td>
</tr>
<tr>
<td>port</td>
<td>Port of the server host</td>
</tr>
<tr>
<td>name</td>
<td>Name of the service instance</td>
</tr>
<tr>
<td>vhost</td>
<td>Name of the messaging server virtual host - a replacement for a <code>name</code> specific to AMQP providers</td>
</tr>
<tr>
<td>username</td>
<td>Server user</td>
</tr>
<tr>
<td>password</td>
<td>Server password</td>
</tr>
</tbody>
</table>

The following is an example output of `ENV[VCAP_SERVICES]`.

```
VCAP_SERVICES=
{
  cleardb: {
    name: "cleardb-1",
    label: "cleardb",
    plan: "spark",
    credentials: {
      name: "ad_c6f4446532610ab",
      hostname: "us-cdbr-east-03.cleardb.com",
      port: "3306",
      username: "b5d435f40dd2b2",
      password: "ebfc00ac",
      uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab",
      jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"
    }
  },
  cloudamqp: {
    name: "cloudamqp-6",
    label: "cloudamqp",
    plan: "lemur",
    credentials: {
      uri: "amqp://ksvyjmiv:IwN6dCdZme0Q400ZPKpu1Y0aL1he8wo8:lemur.cloudamqp.com/ksvyjmiv"
    }
  },
  cloudamqp: {
    name: "cloudamqp-9dbc6",
    label: "cloudamqp",
    plan: "lemur",
    credentials: {
      uri: "amqp://vhuklnxa:91NFxp5uJsaAdTts98vQ1dXXW3Mo7yMyV8lemur.cloudamqp.com/vhuklnxa"
    }
  },
  rediscloud: {
    name: "rediscloud-1",
    label: "rediscloud",
    plan: "20mb",
    credentials: {
      port: "6379",
    }
  }
}
```

Note: Depending on the types of databases you are using, each database might return different credentials.
host: "pub-redis-6379.us-east-1-2.3.ec2.redislabs.com",
password: "1M5zd3qfW9nUyYa"
}
Enabling Service Instance Sharing

Page last updated:

This topic provides information about enabling service instance sharing in managed services.

Overview

Service authors can allow instances of their services to be shared across spaces and orgs within Cloud Foundry. This enables apps running in different spaces and orgs to use the same service instance. Developers with Space Developer permissions in the space where the service instance was created are responsible for sharing, unsharing, updating, and deleting the service instance.

For more information about the developer and administrator tasks related to service instance sharing, see Sharing Service Instances.

Enabling Service Instance Sharing

Service brokers must explicitly enable service instance sharing by setting a flag in their service-level metadata object. This allows service instances, of any service plan, to be shared across orgs and spaces. The "shareable" flag must be set to true in the service-level metadata to enable service instance sharing. If the flag is set to false or is absent, sharing is disabled.

An example catalog is shown below:

```json
{
  "services":{
    "id":"766fa866-a95b-4b12-adff-c11f64cf8fd6",
    "name": "example-service",
    "metadata": {
      "shareable": true
    }
  }
}
```

Binding Permissions Based on Instance Sharing

When a service instance is created in one space and shared into another, developers can bind their apps to the service instance from both spaces.

You may want to have the service broker return credentials with different permissions depending on which space an app is bound from. For example, a messaging service may permit writes from the originating space and only reads from any spaces that the service is shared into.

To determine whether the space of the app is the same as the originating space of the service instance, the service broker can compare the context.space_guid and bind_resource.space_guid fields in the binding request. The context.space_guid field represents the space where the service instance was created, and bind_resource.space_guid represents the space of the app involved in the binding.

Security Considerations

Service authors must consider the following before enabling service instance sharing:

- Service keys can only be generated by users who have access to the space where the service instance was created. This ensures that only developers with access to this space have visibility into where and how many times the service instance is used.
- Consider the impact of giving out excessive permissions for service bindings, especially bindings that originate from spaces that the service instance has been shared into. For example, a messaging service may permit writes from the originating space and only reads from any shared spaces. For more information, see Binding Permissions Based on Instance Sharing.
- You should generate unique credentials for each binding. This ensures that developers can unshare a service instance at any time. Unsharing an instance deletes any service bindings and revokes access for those credentials. Unsharing an instance prevents unauthorized future access from developers and apps that saved the credentials they were previously provided using the service binding.
- Consider the impact of a service instance dashboard being accessed by users of shared service instances. If authenticating through SSO, see Dashboard Single Sign-On.
Application Log Streaming

Page last updated:

By binding an application to an instance of an applicable service, Cloud Foundry will stream logs for the bound application to the service instance.

- Logs for all apps bound to a log-consuming service instance will be streamed to that instance
- Logs for an app bound to multiple log-consuming service instances will be streamed to all instances

To enable this functionality, a service broker must implement the following:

1. In the `catalog` endpoint, the broker must include `requires: syslog_drain`. This minor security measure validates that a service returning a `syslog_drain_url` in response to the `bind` operation has also declared that it expects log streaming. If the broker does not include `requires: syslog_drain`, and the bind request returns a value for `syslog_drain_url`, Cloud Foundry will return an error for the bind operation.

2. In response to a `bind` request, the broker should return a value for `syslog_drain_url`. The syslog URL has a scheme of syslog, syslog-tls, or https and can include a port number. For example:
   ```
   "syslog_drain_url": "syslog://logs.example.com:1234"
   ```

How does it work?

1. Service broker returns a value for `syslog_drain_url` in response to bind
2. Loggregator periodically polls an internal Cloud Controller endpoint for updates
3. Upon discovering a new `syslog_drain_url`, Loggregator identifies the associated app
4. Loggregator streams app logs for that app to the locations specified by the service instances' `syslog_drain_url`

Users can manually configure app logs to be streamed to a location of their choice using User-provided Service Instances. For details, see Using Third-Party Log Management Services.
Route Services

Introduction

Cloud Foundry application developers may wish to apply transformation or processing to requests before they reach an application. Common examples of use cases include authentication, rate limiting, and caching services. Route Services are a kind of Marketplace Service that developers can use to apply various transformations to application requests by binding an application's route to a service instance. Through integrations with service brokers and, optionally, with the Cloud Foundry routing tier, providers can offer these services to developers with a familiar, automated, self-service, and on-demand user experience.

Note: The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also manage route services using Apps Manager. For more information, see the Manage Route Services section of the Managing Apps and Service Instances Using Apps Manager topic.

Architecture

Cloud Foundry supports the following three models for Route Services:

- **Fully-brokered services**
- **Static, brokered services**
- **User-provided services**

In each model, you configure a route service to process traffic addressed to an app.

Fully-Brokered Service

In the fully-Brokered Service model, the CF router receives all traffic to apps in the deployment before any processing by the route service. Developers can bind a route service to any app, and if an app is bound to a route service, the CF router sends its traffic to the service. After the route service processes requests, it sends them back to the load balancer in front of the CF router. The second time through, the CF router recognizes that the route service has already handled them, and forwards them directly to app instances.

The route service can run inside or outside of CF, so long as it fulfills the Service Instance Responsibilities to integrate it with the CF router. A service broker publishes the route service to the CF marketplace, making it available to developers. Developers can then create an instance of the service and bind it to their apps with the following commands:

```bash
cf create-service BROKER-SERVICE-PLAN SERVICE-INSTANCE

cf bind-route-service YOUR-APP-DOMAIN SERVICE-INSTANCE [--hostname HOSTNAME] [--path PATH]
```

Developers configure the service either through the service provider's web interface or by passing arbitrary parameters to their `cf create-service` call through the `-c` flag.
Advantages:

- Developers can use a Service Broker to dynamically configure how the route service processes traffic to specific applications.
- Adding route services requires no manual infrastructure configuration.
- Traffic to apps that do not use the service makes fewer network hops because requests for those apps do not pass through the route service.

Disadvantages:

- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

Static, Brokered Service

In the static, brokered service model, an operator installs a static routing service, which might be a piece of hardware, in front of the load balancer. The routing service runs outside of Cloud Foundry and receives traffic to all apps running in the CF deployment. The service provider creates a service broker to publish the service to the CF marketplace. As with a **fully-brokered service**, a developer can use the service by instantiating it with `cf create-service` and binding it to an app with `cf bind-route-service`.

![Diagram of Static, Brokered Service](image)

In this model, you configure route services on an app-by-app basis. When you bind a service to an app, the service broker directs the routing service to process that app's traffic rather than pass the requests through unchanged.

Advantages:

- Developers can use a Service Broker to dynamically configure how the route service processes traffic to specific applications.
- Traffic to apps that use the route service takes fewer network hops.

Disadvantages:

- Adding route services requires manual infrastructure configuration.
- Traffic to apps that do not use the route service make unnecessary network hops. Requests for all apps hosted by the deployment pass through the route service component.

User-Provided Service

If a route service is not listed in the CF marketplace by a broker, a developer can still bind it to their app as a user-provided service. The service can run anywhere, either inside or outside of CF, but it must fulfill the integration requirements described in **Service Instance Responsibilities**. The service also needs to be reachable by an outbound connection from the CF router.
This model is identical to the *fully-brokered service* model, except without the broker. Developers configure the service manually, outside of Cloud Foundry. They can then create a user-provided service instance and bind it to their application with the following commands, supplying the URL of their route service:

```bash
cf create-user-provided-service SERVICE-INSTANCE -r ROUTE-SERVICE-URL
cf bind-route-service DOMAIN SERVICE-INSTANCE [--hostname HOSTNAME]
```

**Advantages:**

- Adding route services requires no manual infrastructure configuration.
- Traffic to apps that do not use the service makes fewer network hops because requests for those apps do not pass through the route service.

**Disadvantages:**

- Developers must manually provision and configure route services out of the context of Cloud Foundry because no service broker automates these operations.
- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

### Architecture Comparison

The models above require the broker and service instance responsibilities summarized in the following table:

<table>
<thead>
<tr>
<th>Route Services Architecture</th>
<th>Fulfills CF Service Instance Responsibilities</th>
<th>Fulfills CF Broker Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully-Brokered</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Static Brokered</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>User-Provided</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

### Enabling Route Services in Pivotal Cloud Foundry

You configure Route Services for your deployment in the PAS tile, under Settings > Networking. For more information, see Configuring PAS.

### Service Instance Responsibilities

The following applies only when a broker returns `route_service_url` in the bind response.

#### How It Works

Binding a service instance to a route associates the `route_service_url` with the route in the CF router. All requests for the route are proxied to the URL specified by `route_service_url`.

Once a route service completes its function it may choose to forward the request to the originally requested URL or to another location, or it may choose to reject the request; rejected requests will be returned to the originating requestor. The CF router includes a header that provides the originally requested URL, as well as two headers that are used by the router itself to validate the request sent by the route service. These headers are described below in Headers.
Headers

The following HTTP headers are added by the Gorouter to requests forwarded to route services.

X-CF-Forwarded-Url

The X-CF-Forwarded-Url header contains the originally requested URL. The route service may choose to forward the request to this URL or to another.

X-CF-Proxy-Signature

The X-CF-Proxy-Signature header contains an encrypted value which only the Gorouter can decrypt.

The header contains the originally requested URL and a timestamp. When this header is present, the Gorouter will reject the request if the requested URL does not match that in the header, or if a timeout has expired.

X-CF-Proxy-Signature also signals to the Gorouter that a request has transited a route service. If this header is present, the Gorouter will not forward the request to a route service, preventing recursive loops. For this reason, route services should not strip off the X-CF-Proxy-Signature and X-CF-Proxy-Metadata headers.

If the route service forwards the request to a URL different from the originally requested one, and the URL resolves to a route for an application on Cloud Foundry, the route must not have a bound route service or the request will be rejected, as the requested URL will not match the one in the X-CF-Proxy-Signature header.

X-CF-Proxy-Metadata

The X-CF-Proxy-Metadata header aids in the encryption and description of X-CF-Proxy-Signature.

SSL Certificates

When Cloud Foundry is deployed in a development environment, certificates hosted by the load balancer are self-signed, and not signed by a trusted certificate authority. When the route service finishes processing an inbound request and makes a call to the value of X-CF-Forwarded-Url, be prepared to accept the self-signed certificate when integrating with a non-production deployment of Cloud Foundry.

Timeouts

Route services must forward the request to the application route within 60 seconds.

In addition, all requests must respond in 900 seconds.

Broker Responsibilities

Catalog Endpoint

Brokers must include requires: ["route_forwarding"] for a service in the catalog endpoint. If this is not present, Cloud Foundry will not permit users to bind an instance of the service to a route.

Binding Endpoint

When users bind a route to a service instance, Cloud Foundry sends a bind request to the broker, including the route address with bind_resource.route. A route is an address used by clients to reach apps mapped to the route. The broker may return a route_service_url, containing a URL where Cloud Foundry should proxy requests for the route. This URL must have an https scheme, otherwise the Cloud Controller rejects the binding. route_service_url is optional, and not returning this field enables a broker to dynamically configure a network component already in the request path for the route, requiring no change in the CF router.
Example Route Services

- **Logging Route Service**: This route service can be pushed as an app to Cloud Foundry. It fulfills the service instance responsibilities above and logs requests received and sent. It can be used to see the route service integration in action by tailing its logs.
- **Rate Limiting Route Service**: This example route service is a simple Cloud Foundry app that provides rate limiting to control the rate of traffic to an application.
- **Spring Boot Example**: Logs requests received and sent; written in Spring Boot

Tutorial

The following instructions show how to use the **Logging Route Service** described in Example Route Services to verify that when a route service is bound to a route, requests for that route are proxied to the route service.

A video of this tutorial is available on Youtube.

These commands requires the Cloud Foundry Command Line Interface (cf CLI) version 6.16 or later.

1. Push the **Logging Route Service** as an app.

   ```bash
   $ cf push logger
   ```

2. Create a user-provided service instance, and include the route of the **Logging Route Service** you pushed as `route_service_url`. Be sure to use `https` for the scheme.

   ```bash
   $ cf create-user-provided-service mylogger -r https://logger.cf.example.com
   ```

3. Push a sample app like **Spring Music**. By default this creates a route `spring-music.cf.example.com`.

   ```bash
   $ cf push spring-music
   ```

4. Bind the user-provided service instance to the route of your sample app. The `bind-route-service` command takes a route and a service instance; the route is specified in the following example by domain `cf.example.com` and hostname `spring-music`.

   ```bash
   $ cf bind-route-service cf.example.com mylogger --hostname spring-music
   ```

5. Tail the logs for your route service.

   ```bash
   $ cf logs logger
   ```

6. Send a request to the sample app and view in the route service logs that the request is forwarded to it.

   ```bash
   $ curl spring-music.cf.example.com
   ```

Security Considerations

The contract between route services and Gorouter, applicable for **Fully Brokered** and **User Provided** models, enables a Cloud Foundry operator to suggest whether or not requests forwarded from the route service to a load balancer are encrypted. The Cloud Foundry operator makes this suggestion by setting the `router.route_services_recommend_https` manifest property.

This suggestion does not allow the platform to guarantee that a route service obeys the scheme of the `X-Forwarded-Proto` header. If a route service ignores the scheme and downgrades the request to plain text, a malicious actor can intercept the request and use or modify the data within it.

For increased security, follow the recommendations below:

- A load balancer terminating TLS should sanitize and reset based on whether the request it received was encrypted or not. Set the `router.sanitize_forwarded_proto: false` manifest property for Gorouter.
- A load balancer configured for TCP passthrough should sanitize and reset the header based on whether the request it received was encrypted or not. Set the `router.sanitize_forwarded_proto: true` manifest property for Gorouter.
For more information about securing traffic into Cloud Foundry, see [Securing Traffic into Cloud Foundry](#).

**Note:** When a route service is mapped to a route, Gorouter sanitizes the `X-Forwarded-Proto` header once, even though requests pass through Gorouter more than once.

---

### Recommendations for Securing Route Services

To best secure communications through route services, Cloud Foundry operators should do the following:

1. Set the `router.route_services_recommended_https: true` manifest property.

2. Set the `router.disable_http: true` manifest property. Setting this property disables the HTTP listener, forcing all communication to Gorouter to be HTTPS. This assumes all route services will communicate over HTTPS with Gorouter. This causes requests from other clients made to port 80 to be rejected. You should confirm that clients of all applications make requests over TLS.

3. Confirm that route services do not modify the value of the `X-Forwarded-Proto` header.

### Route Service Internal Lookup Considerations

If the **Bypass security checks for route service lookup** option is selected, traffic being sent to a route service will not make a trip back through the Load Balancer. This will reduce the performance penalty of internally hosted route services, but introduces the following security risk for traffic going to externally hosted route services. This vulnerability exists for both externally hosted route services and route services hosted on a separate foundation.

In bypass mode, the router will skip the Load Balancer when it finds an existing route for the route service application. If your route service is hosted outside of the platform, and a space developer registers the route service domain as a private domain on the platform, they are able to map a route to their own application to intercept traffic intended for the route service.

If all of your route services are hosted internally on the same foundation, you are not at risk for this vulnerability, however you would be at risk if externally hosted route services are later configured.

When the option is unchecked, the Router will use DNS to make a new connection to the Load Balancer. The domain name for your internally hosted route services need to: Resolve to your Load Balancer Match the TLS Certificate provided by the Load Balancer. These two steps can be verified by visiting the HTTPS URL of the route service directly and confirming that the application received the request via the `cf logs` output for the route service application.

It is also required that the Certificate Authority for the TLS Certificate provided by the Load Balancer is trusted by the Router. Ensure that TLS Certificate from your Load Balancer is either signed by a well known CA, or the CA has been added to the [Certificate Authorities Trusted by Router and HAProxy](https://rpms.haproxy.org/) field of the Networking pane of the PAS and/or Isolation Segment tiles.

It is strongly recommended that this option be unchecked unless your Load Balancer or Router requires Mutual TLS.

### Considerations for platforms using Mutual TLS

If DNS resolves to a TLS-termination point (your Load Balancer or GoRouter) which is configured to require Client Certificates, then you should leave the **Bypass security checks for route service lookup** checkbox selected. When the option is unchecked, the Router will not be able to provide the necessary Client Certificates. This ability will be added in an upcoming patch release. It is recommended that you follow the mitigation steps below until that patch is available.

**Mitigation Steps**

It is possible to mitigate this issue by preemptively registering the route service domains as private domains in a protected Organization in the platform. This will prevent users from intercepting traffic for the externally hosted route service. This mitigation requires you to repeat this process for any new route service domains being added to the platform. Since route services can be added by any space developer, this may be difficult to manage.

---

© Copyright Pivotal Software Inc, 2013-2019
Supporting Multiple Cloud Foundry Instances

Page last updated:

It is possible to register a service broker with multiple Cloud Foundry instances. It may be necessary for the broker to know which Cloud Foundry instance is making a given request. For example, when using Dashboard Single Sign-On, the broker is expected to interact with the authorization and token endpoints for a given Cloud Foundry instance.

There are two strategies that can be used to discover which Cloud Foundry instance is making a given request.

Routing & Authentication

The broker can use unique credentials and/or a unique url for each Cloud Foundry instance. When registering the broker, different Cloud Foundry instances can be configured to use different base urls that include a unique id. For example:

- On Cloud Foundry instance 1, the service broker is registered with the url `broker.example.com/123`
- On Cloud Foundry instance 2, the service broker is registered with the url `broker.example.com/456`

X-Api-Info-Location Header

All calls to the broker from Cloud Foundry include an `X-Api-Info-Location` header containing the `/v2/info` url for that instance. The `/v2/info` endpoint will return further information, including the location of that Cloud Foundry instance’s UAA.

Support for this header was introduced in cf-release v212.
Logging and Metrics

This documentation describes logging and metrics in Cloud Foundry (CF). It includes topics related to monitoring, event logging, CF data sources, and viewing logs and metrics. It also includes information about the Loggregator system, which aggregates and streams logs and metrics from apps and platform components in Pivotal Application Service (PAS).

Contents

Logging and Metrics

- **Overview of Logging and Metrics**: This topic provides an overview of logging and metrics in CF.
- **Application Logging in Cloud Foundry**: This topic describes log types and their messages. It also explains how to view logs from the cf CLI.
- **Customizing Platform Log Forwarding**: This topic describes how to modify log forwarding rules.
- **Configuring the Metric Registrar**: This topic provides an overview of the Metric Registrar. It also explains how to configure the Metric Registrar for exporting custom app metrics to the Loggregator.
- **Identifying the Source Deployments of Metrics**: This topic describes how to identify which runtime deployment is the source of metrics.

Loggregator

- **Loggregator Architecture**: This topic describes the architecture of the Loggregator system.
- **Loggregator Guide for Cloud Foundry Operators**: The topic provides information about configuring Loggregator to avoid data loss with high volumes of logging and metrics data.
- **Deploying a Nozzle to the Loggregator Firehose**: This topic describes deploying a nozzle application to the Loggregator Firehose.
- **Installing the Loggregator Firehose Plugin for cf CLI**: This topic describes how to use the Loggregator Firehose Plugin for cf CLI to access the output of the Firehose.

See Also

For more information about logging, monitoring, and reporting, see the topics below.

- **Configuring Logging in PAS**: This topic describes how to forward system logs to an external service, scale Loggregator component VMs, and manage app traffic logging.
- **Security Event Logging**: This topic describes how to enable and interpret security event logging for the Cloud Controller, the User Account and Authentication (UAA) server, and CredHub.
- **Monitoring Pivotal Cloud Foundry**: This topic describes how operators can monitor their deployments. It includes information about selecting and configuring a monitoring system.
- **Reporting Instance Usage with Apps Manager**: This topic describes how to retrieve app, task, and service instance usage information.
- **Reporting App, Task, and Service Instance Usage**: This topic describes how to use the cf CLI to retrieve system and org level usage information about apps, tasks, and service instances.
- **Monitoring PCF VMs from Ops Manager**: This topic describes how to check current VM status.
Overview of Logging and Metrics

This topic provides an overview of logging and metrics in Cloud Foundry (CF). It includes information about logs and metrics sources and transport systems. It also lists products for viewing logs and metrics.

Sources of Logs and Metrics

There are two sources of CF logs and metrics:

- CF platform components, such as a Diego Cell, MySQL Server, or Cloud Controller
- Apps and app containers deployed on CF

The table below describes the data included in logs and metrics from each source.

<table>
<thead>
<tr>
<th>Source</th>
<th>Logs Data</th>
<th>Metrics Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform components</td>
<td>Logs from CF components</td>
<td>• Health metrics from BOSH-deployed VMs*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Platform metrics from CF components. For example, cell capacity remaining</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and router throughput.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Metrics for any service tile that self-publishes to the Loggregator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Firehose. For example, Redis and MySQL.</td>
</tr>
<tr>
<td>Apps</td>
<td>Logs from apps **</td>
<td>• App container metrics. For example, CPU, memory, and disk usage.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Custom metrics ***</td>
</tr>
</tbody>
</table>

* For more information about using the BOSH Health Monitor to collect health metrics on VMs, see Configuring a Monitoring System.

** For more information about app logging, see Application Logging in Cloud Foundry.

*** For more information about configuring an application to stream custom metrics to Loggregator, see Metrics Forwarder for PCF.

Transport Systems for Logs and Metrics

The following transport systems deliver logs and metrics from their source to an observability product for viewing:

- **Loggregator**: Loggregator is the transport system for both logs and metrics on apps deployed on CF as well as metrics on CF platform components. For more information about the Loggregator system, including Loggregator architecture and components, see Overview of the Loggregator System.
- **rsyslogd on CF component VMs**: rsyslogd is the transport system for CF component logs. Users can configure rsyslogd to transport component logs to a third-party syslog server.

The table below lists the transport system for logs and metrics on CF platform components and apps.

<table>
<thead>
<tr>
<th>Source</th>
<th>Logs Transport System</th>
<th>Metrics Transport System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform components</td>
<td>rsyslogd on CF component VMs</td>
<td>Loggregator</td>
</tr>
<tr>
<td>Apps</td>
<td>Loggregator</td>
<td>Loggregator</td>
</tr>
</tbody>
</table>

Viewing Logs and Metrics

The table below lists the products and tools for viewing CF logs and metrics.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To view system logs from CF components, configure rsyslogd to transport logs to a third-party product.</td>
<td>You can use the following products or tools to view platform component and VM metrics:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Loggregator Firehose CLI Plugin. See Installing the</td>
</tr>
<tr>
<td>Platform components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Apps</th>
</tr>
</thead>
</table>

You can use the following products or tools to view app logs:
- `cf` CLI `cf logs` command. See [Cloud Foundry CLI Reference Guide](https://docs.cloudfoundry.org/cli/).  
- Apps Manager. See [Managing Apps and Service Instances Using Apps Manager](https://docs.cloudfoundry.org/install/cloudfoundry-management-api.html).
- Syslog forwarding. See [Streaming Application Logs to Log Management Services](https://docs.cloudfoundry.org/loggregator/).  
- Loggregator Firehose CLI Plugin. See [Installing the Loggregator Firehose Plugin for CLI](https://docs.cloudfoundry.org/install/cloudfoundry-management-api.html).

You can use the following products or tools to view app metrics:
- Loggregator Firehose CLI Plugin. See [Installing the Loggregator Firehose Plugin for CLI](https://docs.cloudfoundry.org/install/cloudfoundry-management-api.html).
- PCF Metrics. See [PCF Metrics](https://docs.cloudfoundry.org/metrics).
This topic describes the Loggregator architecture and components. It also describes the Pivotal Cloud Foundry (PCF) components that send BOSH-reported VM metrics to Loggregator.

Overview

Loggregator gathers and streams logs and metrics from user apps in a PCF deployment. It also gathers and streams metrics from PCF components and health metrics from BOSH-deployed VMs. Loggregator allows you to view these logs and metrics through the Loggregator CLI plugins or through a third-party service. For more information, see [loggregator](https://github.com) in GitHub.

The Loggregator architecture includes components for collecting, storing, and forwarding logs and metrics.

Loggregator Architecture and Components

The diagram below shows the architecture of Loggregator, including the PCF components with which it interacts.

- **Loggregator Agent**: Loggregator Agents run on both PCF component VMs and Diego cell VMs. They receive logs and metrics from the apps and PCF components located on those VMs. Loggregator Agents then forward the logs and metrics to Dopplers.
- **Doppler**: Dopplers receive logs and metrics from Loggregator Agents, store them in temporary buffers, and forward them to Traffic Controllers.
- **Traffic Controller**: Traffic Controllers poll Doppler servers for logs and metrics, then translate these messages from the Doppler servers as necessary for external and legacy APIs. The Loggregator Firehose is located on the Traffic Controller.
- **Reverse Log Proxy**: Reverse Log Proxies (RLPs) collect logs and metrics from Dopplers and forward them to Syslog Adapters for syslog drains. Operators can scale up the number of RLPs based on overall log volume.
- **Syslog Adapter**: Syslog Adapters manage connections with and write to syslog drains. Operators can scale the Syslog Adapter component based on the number of syslog drains.
- **Firehose**: The Firehose is a WebSocket endpoint that streams all the event data from a PCF deployment. The data stream includes HTTP events, app logs, container metrics from apps, and metrics from PCF platform components. The Firehose cf CLI plugin allows you to view the output of the Firehose. For more information about the Firehose plugin, see [Installing the Loggregator Firehose Plugin for cf CLI](#).

- **Log Cache**: The Log Cache allows you to view logs and metrics from the Firehose over a specified period of time. The Log Cache includes API endpoints and a CLI plugin to query and filter logs and metrics. To download the Log Cache CLI plugin, see [Cloud Foundry Plugins](#). The Log Cache API endpoints are available by default. For more information about using the Log Cache API, see [Log Cache](#) on GitHub.

- **Nozzles**: Nozzles are programs that consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and to forward it to other apps and services. For more information about creating and deploying nozzles, see [Deploying a Nozzle to the Loggregator](#) Firehose.

### Related BOSH Components

This section describes the PCF components that forward BOSH-reported VM metrics to Loggregator. BOSH-reported VM metrics measure the health of BOSH-deployed VMs on which apps and PCF components are deployed. Loggregator streams BOSH-reported VM metrics through the Firehose.

The following are the PCF components that send BOSH-reported VM metrics to Loggregator:

- **BOSH Agent**: BOSH Agents are located on component VMs and Diego cell VMs. They collect metrics, such as cell capacity remaining, from the VM and forward them to the BOSH Health Monitor.

- **BOSH Health Monitor**: The BOSH Health Monitor receives metrics from the BOSH Agents. It then forwards the metrics to a third-party service or to the BOSH System Metrics Forwarder.

- **BOSH System Metrics Plugin**: This plugin reads health events, such as VM heartbeats and alerts from the BOSH Health Monitor, and streams them to the BOSH System Metrics Server.

- **BOSH System Metrics Server**: The BOSH System Metrics Server streams metrics and heartbeat events to the BOSH System Metrics Forwarder over gRPC.

- **BOSH System Metrics Forwarder**: The BOSH System Metrics Forwarder is located on the Loggregator Traffic Controller. It forwards heartbeat events from the BOSH System Metrics Server as envelopes to Loggregator through a Loggregator Agent.
Loggregator Guide for Cloud Foundry Operators

Page last updated:

This topic contains information for Cloud Foundry deployments operators about how to configure the Loggregator system to avoid data loss with high volumes of logging and metrics data.

Loggregator Message Throughput and Reliability

For determining the message throughput and reliability rates of your Loggregator system, see the section below.

Measuring Message Throughput

To measure the message throughput of the Loggregator system, you can monitor the total number of egress messages from all Metrons on your platform using the `metron.egress` metric.

If you do not use a monitoring platform, you can follow the instructions below to measure the overall message throughput of your Loggregator system:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI) with your admin credentials:
   ```
   $ cf login
   ```

2. Install the Cloud Foundry Firehose plugin.

3. Install Pipe Viewer:
   ```
   $ apt-get install pv
   ```

4. Run the following command:
   ```
   $ cf nozzle -n | pv -l -i 10 -r > /dev/null
   ```

Measuring Message Reliability

To measure the message reliability rate of your Loggregator system, you can run black-box tests. If you want to use this method, see the open-source `cf-logmon` app and the configuration instructions provided in the README.md file.

Scaling Loggregator

Most Loggregator configurations use preferred resource defaults. For more information about customizing these defaults and planning the capacity of your Loggregator system, see Key Capacity Scaling Indicators.

Scaling Nozzles

Nozzles are programs that consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and to forward it to other apps and services. You can scale a nozzle using the subscription ID specified when the nozzle connects to the Firehose. If you use the same subscription ID on each nozzle instance, the Firehose evenly distributes data across all instances of the nozzle.

For example, if you have two nozzle instances with the same subscription ID, the Firehose sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the Firehose sends one-third of the data to each instance.

If you want to scale a nozzle, the number of nozzle instances should match the number of Traffic Controller instances:

```
Number of nozzle instances = Number of Traffic Controller instances
```
Stateless nozzles should handle scaling gracefully. If a nozzle buffers or caches the data, the nozzle author must test the results of scaling the number of nozzle instances up or down.

### Slow Nozzle Alerts

The Traffic Controller alerts nozzles if they consume events too slowly. If a nozzle falls behind, Loggregator alerts the nozzle in two ways:

- **TruncatingBuffer** alerts: If the nozzle consumes messages more slowly than they are produced, the Loggregator system may drop messages. In this case, Loggregator sends the log message, **TB: Output channel too full. Dropped N messages**, where N is the number of dropped messages. Loggregator also emits a CounterEvent with the name `doppler_proxy.slow_consumer`. The nozzle receives both messages from the Firehose, alerting the operator to the performance issue.

### Forwarding Logs to an External Service

You can configure Pivotal Application Service (PAS) to forward log data from apps to an external aggregator service. Using Log Management Services explains how to bind apps to the external service and configure it to receive logs from Pivotal Application Service (PAS).

### Log Message Size Constraints

When a Diego Cell emits app logs to Metron, Diego breaks up log messages greater than approximately 60 KiB into multiple envelopes.
Deploying a Nozzle to the Loggregator Firehose

Page last updated:

This topic describes deploying a nozzle app to the Cloud Foundry (CF) Loggregator Firehose. For more information about nozzles and the Loggregator Firehose, see Loggregator Architecture. The Cloud Foundry Loggregator team created an example nozzle application for use with this tutorial.

The procedure described below deploys this example nozzle to the Firehose of a Cloud Foundry installation deployed locally with BOSH Lite. For more information about BOSH Lite, see the BOSH Lite GitHub repository.

To lighten the load on custom nozzles you develop, you can request Firehose subscriptions that emit only whitelisted metrics. For examples, see rlpreader and rlpreader in the loggregator-tools repository, and see the V2 Subscriptions page of the loggregator-release repository for more information.

Prerequisites

Before you deploy a nozzle to the Loggregator Firehose, you must have the following:

- BOSH CLI. See BOSH CLI installed locally.
- Spiff installed locally and added to the load path of your shell. See Spiff on GitHub.
- BOSH Lite deployed locally using VirtualBox. See BOSH Lite on GitHub.
- A working Cloud Foundry deployment, including Loggregator, deployed with your local BOSH Lite. This serves as the source of data. See Deploying Cloud Foundry using BOSH Lite, or use the provision_cf script included in the BOSH Lite release. See BOSH Lite on GitHub.

Note: Deploying Cloud Foundry can take several hours depending on your internet bandwidth, even when using the automated provision_cf script.

Step 1: Download Cloud Foundry BOSH Manifest

To download the BOSH manifest, do the following:

1. Run `bosh -e MY-ENV deployments` to identify the names of all deployments in the environment you specify. Replace `MY-ENV` with the alias you set for your BOSH Director. For example:

```
$ bosh -e dev deployments
Using environment '192.168.15.4' as client 'admin'
Name               Release(s)  Stemcell(s)  Team(s)  Cloud Config
1                   capu1.21.0    cf-mysql/34       diego/1.8.1
                           cf-smoke-tests/11  etcd/94
                           cfunit/3-smooth/1.52.0
cflinuxfs3-rootfs/1.52.0 consul/155
                           nginx/1.14.0
diego/1.8.1            garden-runc/1.2.0
                           loggregator/78
                           mysql/0.6.0
                           routing/0.145.0
                           statsd-injector/1.0.20
                           uaa/25
                           bosh-warden-boshlite-abantu-trusty-go_agent/3363.9 - latest
                           bosh-warden-boshlite-abantu-trusty-go_agent/3363.9 - latest
   2 deployments
   Succeeded
```

2. Run `bosh -e MY-ENV -d MY-DEPLOYMENT manifest > MY-MANIFEST.yml` to download and save the current BOSH deployment manifest. Replace `MY-ENV` with your BOSH Director alias, `MY-DEPLOYMENT` with the deployment name from the output of the previous step, and `MY-MANIFEST.yml` with a name you choose for the saved manifest file. You need this manifest to locate information about your databases.

```
$ bosh -e dev -d cf-example manifest > cf.yml
```
Step 2: Add UAA Client

You must authorize the example nozzle as a UAA client for your CF deployment. To do this, add an entry for the example nozzle as a client for UAA under the `properties` key in your CF deployment manifest YAML file. You must enter the example nozzle object in the correct location in the manifest, and with the correct indentation.

To add the nozzle as a UAA client for your deployment, do the following:

1. Open the deployment manifest in a text editor.

2. Locate the left-aligned `properties` key.

3. Under the `properties` key, locate `uaa` at the next level of indentation.

4. Under the `uaa` key, locate the `clients` key at the next level of indentation.

5. Enter properties for the `example-nozzle` at the next level of indentation, exactly as shown below. The `...` in the text below indicate other properties that may populate the manifest at each level in the hierarchy.

   ```yaml
   properties:
     ...
     uaa:
       ...
       clients:
         ...
         example-nozzle:
           access-token-validity: 1209600
           authorized-grant-types: client_credentials
           override: true
           secret: example-nozzle
           authorities: oauth.login,doppler.firehose
   ...
   ...
   ```

6. Save the deployment manifest file.

Step 3: Redeploy Cloud Foundry

To deploy Cloud Foundry with BOSH, run the following command:

```bash
bosh -e MY-ENV deploy
```

WHERE `MY-ENV` is the alias you set for your BOSH Director.

For example:
Step 4: Clone Example Release

The Cloud Foundry Loggregator team created an example nozzle app for use with this tutorial.

To clone the example nozzle release, do the following:

1. Run `git clone` to clone the main release repository from GitHub.

   ```bash
   $ git clone https://github.com/cloudfoundry-incubator/example-nozzle-release.git
   Cloning into 'example-nozzle-release'...
   ...
   ```

2. Navigate to the `example-nozzle-release` directory.

   ```bash
   $ cd example-nozzle-release
   ...
   ```

3. Run `git submodule update --init --recursive` to update all of the included submodules.

   ```bash
   $ git submodule update --init --recursive
   Submodule 'src/github.com/cloudfoundry-incubator/example-nozzle' (git@github.com:cloudfoundry-incubator/example-nozzle.git) registered for path 'src/github.com/cloudfoundry-incubator/example-nozzle'
   Submodule 'src/github.com/cloudfoundry-incubator/uaago' (git@github.com:cloudfoundry-incubator/uaago.git) registered for path 'src/github.com/cloudfoundry-incubator/uaago'
   ...
   Cloning into 'src/github.com/cloudfoundry-incubator/example-nozzle'...
   ...
   ```
Step 5: Prepare Nozzle Manifest

To prepare the nozzle deployment manifest, do the following:

1. In the `example-nozzle-release` directory, navigate to the `templates` directory.

   ```
   $ cd templates
   ```

   Within this directory, examine the two YAML files. `bosh-lite-stub.yml` contains the values used to populate the missing information in `template.yml`. By combining these two files we create a deployment manifest for our nozzle.

2. Create a `tmp` directory for the compiled manifest.

3. Use Spiff to compile a deployment manifest from the template and stub, and save this manifest.

   ```
   $ spiff merge templates/template.yml templates/bosh-lite-stub.yml > tmp/manifest_bosh_lite.yml
   ```

4. Run `bosh -e MY-ENV deployments` to identify the names of all deployments in the environment you specify. Replace `MY-ENV` with the alias you set for your BOSH Director.

5. Run `bosh -e MY-ENV env --column=uuid` to obtain your BOSH Director UUID. Replace `MY-ENV` with the alias you set for your BOSH Director. For example:

   ```
   $ bosh -e dev env --column=uuid
   ```

6. In the compiled nozzle deployment manifest, locate the `director uuid` property. Replace `PLACEHOLDER-DIRECTOR-UUID` with your BOSH Director UUID.

   ```
   compilation:
   cloud_properties:
   name: default
   network: example-nozzle-net
   reuse_compilation_vms: true
   workers: 1
   director uuid: PLACEHOLDER-DIRECTOR-UUID # replace this
   ```

   **Note:** If you do not want to see the complete deployment procedure, run the following command to automatically prepare the manifest:

   ```
   scripts/make_manifest_spiff_bosh_lite
   ```

Step 6: Create Nozzle BOSH Release

To create a nozzle BOSH release, run the following command:

```
bosh -e MY-ENV create-release --name RELEASE-NAME
```

WHERE:

- `MY-ENV` is the alias you set for your BOSH Director.
- `RELEASE-NAME` is `example-nozzle` to match the UAA client that you created in the CF deployment manifest.

For example:

```
$ bosh -e dev create-release --name example-nozzle
Syncing blobs...
... 
```

Step 7: Upload Nozzle BOSH Release

Upload the nozzle BOSH release that you created as part of [Step 6: Create Nozzle BOSH Release](#).
To upload the BOSH release, run the following command:

```bash
bosh -e MY-ENV upload-release
```

WHERE `[MY-ENV]` is the alias you set for your BOSH Director.

For example:

```
$ bosh -e dev upload-release

Copying packages
----------------
example-nozzle
golang1.7

Copying jobs
------------
example-nozzle

Generated /var/folders/4n/qs1rjbmd1c3g78in3_06j6r000g/T/d20151009-71219-17a5m49/d20151009-71219-en0928/release.tgz
Release size: 59.2M

Verifying release...
...
Release info
Name: nozzle-test
Version: 0+dev.2

Packages
- example-nozzle (b0944095eb5a332e9be2a2db1b8e8b75598f)
- golang1.7 (b68dc9557cf296cb21e577c51b97c2584a5154b)

Jobs
- example-nozzle (112e016ee91e8b2d02a2239e58edc1e18e0360d35)

License
- none

Uploading release
```

Step 8: Deploy Nozzle

To deploy the nozzle, run the following command:

```bash
bosh -e MY-ENV deploy
```

WHERE `[MY-ENV]` is the alias you set for your BOSH Director.

For example:

```
$ bosh -e dev deploy

Deploying
---------
Are you sure you want to deploy? (type 'yes' to continue):yes
```

Step 9: View Nozzle Output

The example nozzle outputs all of the data originating coming from the Firehose to its log files. To view this data, SSH into the example-nozzle VM and examine the logs.
To view nozzle output, do the following:

1. Run `bosh -e MY-ENV ssh` to access the nozzle VM at the IP configured in the nozzle’s manifest template stub `/templates/bosh-lite-stub.yml`. Replace `MY-ENV` with the alias you set for your BOSH Director. For example:

   ```
   $ bosh -e dev ssh example-nozzle
   Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.19.0-25-generic x86_64)
   Documentation: https://help.ubuntu.com/
   Last login: Wed Sep 23 21:29:50 2015 from 192.0.2.1
   ```

2. Use the `cat` command to output the `stdout` log file.

   ```
   $ cat /var/vcap/sys/log/example-nozzle/example-nozzle.stdout.log
   ```
Installing the Loggregator Firehose Plugin for cf CLI

Page last updated:

The Loggregator Firehose plugin for the Cloud Foundry Command Line Interface (cf CLI) allows Cloud Foundry (CF) administrators to access the output of the Loggregator Firehose. The output of the Firehose includes logs and metrics from apps deployed on CF as well as metrics from CF platform components. For more information about the Firehose, see [Overview of the Loggregator System](#).

For more information about using plugins with the cf CLI, see [Using cf CLI Plugins](#).

Prerequisites

- Administrator access to the Cloud Foundry deployment that you want to monitor
- Cloud Foundry Command Line Interface (cf CLI) 6.12.2 or later

Refer to the [Installing the cf CLI](#) topic for information about downloading, installing, and uninstalling the cf CLI.

Install the Plugin

1. Run `cf add-plugin-repo REPOSITORY-NAME URL` to add the CF Community plugin repository to your cf CLI plugins.

   ```
   $ cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
   ```

2. Run `cf install-plugin -r PLUGIN-REPOSITORY PLUGIN-NAME` to install the Firehose plugin from the CF Community plugin repository.

   ```
   $ cf install-plugin -r CF-Community "Firehose Plugin"
   ```

View the Firehose

Run `cf nozzle --debug` to view the streaming output of the Firehose, which includes logging events and metrics from CF system components. For more information about logging and metrics in CF, see [Overview of the Loggregator System](#).

```
$ cf nozzle --debug
```

**Note:** You must be logged in as a Cloud Foundry administrator to access the Firehose.

Uninstall the Plugin

Run `cf plugins` to see a list of installed plugins.

```
$ cf plugins
Listing Installed Plugins...
OK
Plugin Name  Version  Command Name  Command Help
FirehosePlugin  0.6.0  nozzle  Command to print out messages from the firehose
```

Run `cf uninstall-plugin PLUGIN-NAME` to uninstall the plugin.

```
$ cf uninstall-plugin FirehosePlugin
```
Customizing Platform Log Forwarding

Page last updated:

You can configure Pivotal Application Service (PAS) to forward logs to remote endpoints using the syslog protocol defined in RFC 5424. For more information about enabling log forwarding, see the Configure System Logging section of Configuring PAS.

PAS annotates forwarded messages with structured data. This structured data identifies the originating BOSH Director, deployment, instance group, availability zone, and instance ID. All logs forwarded from BOSH jobs have their PRI set to 14, representing Facility: user-level messages and Warning: warning conditions, as defined in RFC 5424 Section 6.2.1. This PRI value may not reflect the originally intended PRI of the log.

Logs forwarded from other sources, such as kernel logs, retain their original PRI value.

The following table describes the log line Structured Data:

<table>
<thead>
<tr>
<th>Structured Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTERPRISE_NUMBER</td>
<td>Cloud Foundry’s private enterprise number, 47450, as defined in RFC 5424 Section 7.2.2.</td>
</tr>
<tr>
<td>DIRECTOR</td>
<td>The name of the BOSH Director managing the deployment.</td>
</tr>
<tr>
<td>DEPLOYMENT</td>
<td>The name of the BOSH deployment.</td>
</tr>
<tr>
<td>INSTANCE_GROUP</td>
<td>The name of the BOSH instance_group.</td>
</tr>
<tr>
<td>AVAILABILITY_ZONE</td>
<td>The name of the BOSH availability zone.</td>
</tr>
<tr>
<td>ID</td>
<td>The BOSH GUID.</td>
</tr>
</tbody>
</table>

Log lines use the format below:

```
<$PRI>$VERSION $TIMESTAMP $HOST $APP_NAME $PROC_ID $MSG_ID [instance=ENTERPRISE_NUMBER director=DIRECTOR deployment=DEPLOYMENT group=INSTANCE_GROUP az=AVAILABILITY_ZONE id=ID] $MESSAGE
```

Example log messages:

```
<14>1 2017-01-25T13:25:03.18377Z 192.0.2.10 etcd - - [instance@47450 director=test-env deployment=cf-c42ae2c4dfb6f676b6c27 group=diego_database az=us-west1-a id=83bd66e5-3dfd-44b7-8d6a-0588eb787687] [INFO] the leader is [https://diego-database-0.etcd.service.cf.internal:4001]
```

Modify Which Logs PAS Forwards

When you enable log forwarding, PAS forwards all log lines written to the /var/vcap/sys/log directories on all Cloud Foundry virtual machines (VMs) to your configured External Syslog Aggregator endpoint by default.

You can configure PAS to forward a subset of logs instead of forwarding all logs as follows.

1. In the PAS tile, select System Logging.

2. In the Custom rsyslog Configuration textbox, enter a custom syslog rule. See the example custom syslog rules below.

3. Click Save.

The custom syslog rules shown below are written in RainerScript. The custom rules are inserted before the rule that forwards logs. The stop command, stop, prevents logs from reaching the forwarding rule. This filters out these logs.

© Copyright Pivotal Software Inc, 2013-2019
Logs filtered out before forwarding remain on the local disk, where the BOSH job originally wrote them. These logs remain on the local disk only until BOSH Director recreates the VMs. You can access these logs from Ops Manager or through SSH.

**Note:** PAS requires a valid custom rule to forward logs. If your custom rule contains syntax errors, PAS forwards no logs.

### Forward Only Logs From a Certain Job

This rule filters out logs unless they originate from the `uaa` job:

```plaintext
if ($app-name != "uaa") then stop
```

### Exclude Logs With Certain Content

This rule filters out logs that contain “DEBUG” in the body.

```plaintext
if ($msg contains "DEBUG") then stop
```

**Note:** The above example contains “DEBUG” in the message body. Not all logs intended for debugging purposes contain the string “DEBUG” in the message body.
Configuring the Metric Registrar

This topic describes the Metric Registrar for Pivotal Application Service (PAS). It also includes information about enabling and configuring the Metric Registrar.

Overview

The Metric Registrar allows app developers to export custom app metrics in a format that Loggregator can consume. App developers can then use the custom metrics to monitor apps with PCF Metrics and configure autoscaling rules with PCF Autoscaler.

App developers can export custom metrics to Loggregator by configuring their apps in one of the following ways:

- **Metrics Endpoint**—Publish and register a Prometheus Exposition metrics endpoint to an app. The Metric Registrar will then poll this endpoint every 35 seconds and convert the metrics found in the response to Loggregator metrics.

- **Structured Log**—Modify your app to emit metrics using a specific JSON or DogStatsD format. The Metric Registrar then converts all matching log messages into Loggregator metrics or events.

For more information about installing the Metric Registrar Plugin and registering your app, see [Emitting Custom App Metrics to the Metric Registrar](#).

For more information about the components and products mentioned, see the following:
- [Loggregator](#)
- [PCF Metrics](#)
- [PCF Autoscaler](#)

Architecture

The following diagram illustrates how the Metric Registrar sends your custom app metrics to Loggregator. The components of the Metric Registrar are as follows:

- The `cf` CLI plugin
- The `metric_registrar_endpoint_worker` and `metric_registrar_log_worker` jobs running on the Doppler VM of the PAS deployment
- The `metric_registrar_orchestrator` and `metric_registrar_smoke_test` jobs running on the Clock Global VM of the PAS deployment

Click the image for a larger representation.
Configure the Metric Registrar

The PAS tile includes a Metric Registrar configuration pane. You can configure the Metric Registrar from this pane in the following ways:

- Enable
- Edit default scraping interval
- Add blacklisted tags

Enable the Metric Registrar

The PAS tile does not deploy the Metric Registrar by default.

To enable the Metric Registrar, do the following:

1. In the PAS tile, click Metric Registrar.
2. Select the Enable Metric Registrar checkbox.

Edit Default Scraping Interval

The scraping interval defines how often the Metric Registrar polls custom metric endpoints. The default is 35 seconds.

To edit the Metric Registrar scraping interval, do the following:

1. In the PAS tile, click Metric Registrar.
2. Edit the Endpoint Scraping Interval field.

Add Blacklisted Tags

To prevent the Metric Registrar from consuming the value of a metric or event tag, you can add the tag to the Blacklisted Tags field. For example, if you tag your metrics with a customer_id, you may want to add customer_id to the list of blacklisted tags. The field accepts a comma-separated list.

By default, the following tags are blacklisted to prevent interference with other products like PCF Metrics that use and rely on such tags:

- deployment
- job
- index
- id

To prevent the Metric Registrar from consuming the value of a metric or event tag, do the following:

1. In the PAS tile, click Metric Registrar.
2. Add the desired tag to the Blacklisted Tags field.
Registering Custom App Metrics

This topic describes how to install the Metric Registrar CLI plugin and emit custom app metrics to the Metric Registrar by registering your app as a metric source.

Overview

Registering your app as a metric source allows you to see your custom metrics in PCF Metrics and configure autoscaling rules with PCF Autoscaler.

For more information, see PCF Metrics and PCF Autoscaler.

Install the Plugin

To install the Metric Registrar CLI plugin, do the following:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI).
2. Run the following command:

   ```bash
   cf install-plugin -r CF-Community "metric-registrar"
   ```

Register Your App

To register your app as a metric source, do one of the following:

- **Register a public endpoint**: Register a public endpoint for custom metrics to be parsed and emitted to Loggregator. See Register a Public Metrics Endpoint.
- **Register a structured log format**: Register a structured log format that can be emitted to Loggregator. See Register a Structured Log Format.

**Note**: If you are migrating from and manually send data to Metrics Forwarder for PCF, Pivotal recommends registering a structured log format. See Register a Structured Log Format.

Register a Public Metrics Endpoint

The Metric Registrar supports custom metrics created with the open-source tool Prometheus. Prometheus uses a response format common in many open-source libraries. It also provides several out-of-the-box metrics. For more information about Prometheus, see prometheus.io.

For examples of apps that use Prometheus to publish metrics to an endpoint, see metric-registrar-examples in GitHub.

**Note**: These endpoints must be public for the Metric Registrar to retrieve the data. If you do not want to expose public endpoints for your app, see Register a Structured Log Format below.

Prerequisites

Before registering a public metrics endpoint, you must do the following:

- For all Spring apps, update the `application.yml` file to include one or more Prometheus endpoints. For example:
For all Spring apps, update the security configuration file to permit access to the Prometheus endpoints. For an example, see metric-registrar-examples in GitHub.

Register an Endpoint

To register a public metrics endpoint for an app, do the following:

1. Log in to the cf CLI.
2. For each Prometheus endpoint in your app, run the following command to register the endpoint as a metric source:

   ```bash
   cf register-metrics-endpoint APP-NAME PATH
   ```

   Where:
   - `APP-NAME` is the name of the app.
   - `PATH` is the path to the Prometheus endpoint.

Register a Structured Log Format

The Metric Registrar supports metrics emitted in JSON or DogStatsD formats. For more information about these formats, see the JSON and DogStatsD sections below.

To register your app as a metric source, do the following:

1. Log in to the cf CLI.
2. Run the following command:

   ```bash
   cf register-log-format APP-NAME FORMAT
   ```

   Where:
   - `APP-NAME` is the name of the app.
   - `FORMAT` is either `json` or `DogStatsD`.

JSON

The table below shows the supported JSON format for event, gauge, and counter log types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>{</td>
</tr>
</tbody>
</table>
|        |   "type": "event",
|        |   "title": "title",
|        |   "body": "body",
|        |   "tags": { |
|        |     "tag1": "tag value"
|        |   }      |
|        | }       |
|        | {      |
|        |   "type": "gauge",
|        | }       |
DogStatsD

The table below shows the supported DogStatsD format for event, gauge, and counter log types. It also lists the supported fields. For more information about DogStatsD, see the [DogStatsD](#) topic in the Datadog documentation.

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Supported Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>`{_e{title.length,text.length}:title</td>
<td>text</td>
</tr>
<tr>
<td>Gauges</td>
<td>`{gauge.name:value</td>
<td>g}@sample_rate</td>
</tr>
<tr>
<td>Counters</td>
<td>`{counter.name:value</td>
<td>c}@sample_rate</td>
</tr>
</tbody>
</table>
Identifying the Source Deployment of Metrics

This topic describes operators can identify which runtime deployment the metrics in their Pivotal Cloud Foundry (PCF) foundation are coming from. For example, an operator may want to know whether metrics are coming from Pivotal Application Service (PAS) or PCF Isolation Segment.

Tile GUIDs

As the value for the deployment tag, metrics use cf- and then the GUID for your PAS tile. With a GUID, you can uniquely identify your metrics by tile.

Human-friendly Metadata

PAS and other runtime tiles tag metrics with additional metadata to help operators better parse the metrics coming from their different deployments. Also, downstream monitoring products, such as PCF Healthwatch, use this metadata to display human readable names.

The tags are as follows:

- **product**: The value of this tag is always Pivotal Application Service for the PAS tile. The tags for other products are: PCF Isolation Segment, PCF Small Footprint, Pivotal Application Service for Windows 2012R2, and Pivotal Application Service for Windows.
- **system_domain**: The value of this tag corresponds to what you set in the System Domain field in the Domains tab of a given tile.
- **placement_tag**: The value of this tag is always null for PAS. However, for PAS for Windows and PCF Isolation Segment tiles, you can configure this value using the Segment Name field in the Application Containers pane. An Operator can display capacity and other relevant metrics using the placement_tag name. This makes it easier to reason about the importance of a given segment when issues arise.

These tags are properties of the metron agent running on each VM in a deployment.
Troubleshooting and Diagnostics

Navigate these topics to troubleshoot and diagnose issues you may encounter when installing or using Pivotal Cloud Foundry (PCF).

Table of Contents

- Diagnosing Problems in PCF
- Troubleshooting Problems in PCF
- Advanced Troubleshooting with the BOSH CLI
- Troubleshooting Slow Requests in Cloud Foundry
- Troubleshooting TCP Routing
- Recovering MySQL from PAS Downtime
- Troubleshooting BBR
- Troubleshooting PCF on Azure
- Troubleshooting PCF on GCP
- Troubleshooting Ops Manager for VMware vSphere
- Troubleshooting Windows Cells
- Running mysql-diag
Diagnosing Problems in PCF

Page last updated:

This guide provides help with diagnosing issues encountered during a Pivotal Cloud Foundry (PCF) installation.

Besides whether products install successfully or not, an important area to consider when diagnosing issues is communication between VMs deployed by Pivotal Cloud Foundry. Depending on what products you install, communication takes the form of messaging, routing, or both. If they go wrong, an installation can fail. For example, in an Pivotal Application Service (PAS) installation the PCF VM tries to push a test application to the cloud during post-installation testing. The installation fails if the resulting traffic cannot be routed to the HA Proxy load balancer.

Viewing the Debug Endpoint

The debug endpoint is a web page that provides information useful for diagnostics. If you have superuser privileges and can view the Ops Manager Installation Dashboard, you can access the debug endpoint.

- In a browser, open the URL: [https://OPS-MANAGER-FQDN/debug](https://OPS-MANAGER-FQDN/debug)

The debug endpoint offers three links:

- **Files** allows you to view the YAML files that Ops Manager uses to configure products that you install. The most important YAML file, `installation.yml`, provides networking settings and describes microbosh. In this case, microbosh is the VM whose BOSH Director component is used by Ops Manager to perform installations and updates of PAS and other products.

- **Components** describes the components in detail.

- **Rails log** shows errors thrown by the VM where the Ops Manager web application (a Rails application) is running, as recorded in the `production.log` file. See the next section to learn how to explore other logs.

Logging Tips

**Identifying Where to Start**

This section contains general tips for locating where a particular problem is called out in the log files. Refer to the later sections for tips regarding specific logs (such as those for PAS Components).

- Start with the largest and most recently updated files in the job log
- Identify logs that contain ‘err’ in the name
- Scan the file contents for a “failed” or “error” string

**Viewing Logs for PAS Components**

To troubleshoot specific PAS components by viewing their log files, browse to the Ops Manager interface and follow the procedure below.

1. In Ops Manager, browse to the PAS Status tab. In the Job column, locate the component of interest.
2. In the Logs column for the component, click the download icon.
3. Browse to the PAS Logs tab.

4. Once the zip file corresponding to the component of interest moves to the Downloaded list, click the linked file path to download the zip file.

5. Once the download completes, unzip the file.

The contents of the log directory vary depending on which component you view. For example, the Diego cell log directory contains sub-directories for the metron_agent, rep, monit, and garden processes. To view the standard error stream for garden, download the Diego cell logs and open

diego.0.job > garden >
garden.stderr.log

Viewing Web Application and BOSH Failure Logs in a Terminal Window

You can obtain diagnostic information from the Operations Manager by logging in to the VM where it is running. To log in to the Operations Manager VM, you need the following information:

- The IP address of the PCF VM shown in the Settings tab of the Ops Manager Director tile.
- Your import credentials. Import credentials are the username and password used to import the PCF .ova or .ovf file into your virtualization system.

Complete the following steps to log in to the Operations Manager VM:

1. Open a terminal window.
Run `ssh IMPORT-USERNAME@PCF-VM-IP-ADDRESS` to connect to the PCF installation VM.

Enter your import password when prompted.

Change directories to the home directory of the web application:
```
cd /home/tempest-web/tempest/web/
```

You are now in a position to explore whether things are as they should be within the web application.

You can also verify that the `microbosh` component is successfully installed. A successful MicroBOSH installation is required to install PAS and any products like databases and messaging services.

Change directories to the BOSH installation log home:
```
cd /var/tempest/workspaces/default/deployments/micro
```

You may want to begin by running a tail command on the current log:
```
cd /var/tempest/workspaces/default/deployments/micro
```

If you are unable to resolve an issue by viewing configurations, exploring logs, or reviewing common problems, you can troubleshoot further by running BOSH diagnostic commands with the BOSH Command Line Interface (CLI).

**Note:** Do not manually modify the deployment manifest. Operations Manager will overwrite manual changes to this manifest. In addition, manually changing the manifest may cause future deployments to fail.

### Viewing the VMs in Your Deployment

To view the VMs in your PCF deployment, perform the following steps specific to your IaaS.

**Amazon Web Services (AWS)**

1. Log in to the [AWS Console](#).
2. Navigate to the EC2 Dashboard.
3. Click Running Instances.
4. Click the gear icon in the upper right.
5. Select the following: job, deployment, director, index.
6. Click Close.

**OpenStack**

1. Install the [novaclient](#).
2. Point novaclient to your OpenStack installation and tenant by exporting the following environment variables:

   ```
   $ export OS_AUTH_URL=YOUR_KEYSTONE_AUTH_ENDPOINT
   $ export OS_TENANT_NAME=TENANT_NAME
   $ export OS_USERNAME=USERNAME
   $ export OS_PASSWORD=PASSWORD
   
   $ nova list --fields metadata
   ```

**vSphere**

1. Log into vCenter.
2. Select Hosts and Clusters.

3. Select the top level object that contains your PCF deployment. For example, select Cluster, Datastore or Resource Pool.

4. In the top tab, click Related Objects.

5. Select Virtual Machines.

6. Right click on the Table heading and select Show/Hide Columns.

7. Select the following boxes: job, deployment, director, index.

Viewing Apps Manager Logs in a Terminal Window

The [Apps Manager](#) provides a graphical user interface to help manage organizations, users, applications, and spaces.

When troubleshooting Apps Manager performance, you might want to view the Apps Manager application logs. To view the Apps Manager application logs, follow these steps:

1. Run `cf login -a api.MY-SYSTEM-DOMAIN -u admin` from a command line to log in to PCF using the UAA Administrator credentials. In Pivotal Ops Manager, refer to PAS Credentials for these credentials.

   ```
   $ cf login -a api.example.com -u admin
   API endpoint: api.example.com
   Password>******
   Authenticating...
   OK
   ```

2. Run `cf target -o system -s apps-manager` to target the system org and the apps-manager space.

   ```
   $ cf target -o system -s apps-manager
   ```

3. Run `cf logs apps-manager` to tail the Apps Manager logs.

   ```
   $ cf logs apps-manager
   Connected, tailing logs for app apps-manager in org system / space apps-manager as admin...
   ```

Changing Logging Levels for the Apps Manager

The Apps Manager recognizes the `LOG_LEVEL` environment variable. The `LOG_LEVEL` environment variable allows you to filter the messages reported in the Apps Manager log files by severity level. The Apps Manager defines severity levels using the Ruby standard library Logger class.

By default, the Apps Manager `LOG_LEVEL` is set to info. The logs show more verbose messaging when you set the `LOG_LEVEL` to debug.

To change the Apps Manager `LOG_LEVEL`, run `cf set-env apps-manager LOG_LEVEL` with the desired severity level.

```
$ cf set-env apps-manager LOG_LEVEL debug
```

You can set `LOG_LEVEL` to one of the six severity levels defined by the Ruby Logger class:

- **Level 5**: unknown – An unknown message that should always be logged
- **Level 4**: fatal – An unhandleable error that results in a program crash
- **Level 3**: error – A handleable error condition
- **Level 2**: warn – A warning
- **Level 1**: info – General information about system operation
- **Level 0**: debug – Low-level information for developers

Once set, the Apps Manager log files only include messages at the set severity level and above. For example, if you set `LOG_LEVEL` to final, the log includes final and unknown level messages only.
Analyzing Disk Usage on Containers and Diego Cell VMs

To obtain disk usage statistics by Diego Cell VMs and containers, see Examining GrootFS Disk Usage.
Troubleshooting Problems in PCF

This guide provides help with resolving issues encountered during a Pivotal Cloud Foundry (PCF) installation.

Retrying the Deployment

Although an install or update can fail for many reasons, the system is self-healing, and can often automatically correct or work around hardware or network faults.

Click Install or Review Pending Changes, then Apply Changes again, and the system may resolve a problem on its own.

Some failures only produce generic errors like Exited with 1. In cases like this, where a failure is not accompanied by useful information, click Install or Review Pending Changes, then Apply Changes to retry.

When the system does provide informative evidence, review the Common Problems section at the end of this guide to see if your problem is covered there.

Common Issues

Compare evidence that you have gathered to the descriptions below. If your issue is covered, try the recommended remediation procedures.

BOSH Does Not Reinstall

You might want to reinstall BOSH for troubleshooting purposes. However, if PCF does not detect any changes, BOSH does not reinstall. To force a reinstall of BOSH, select BOSH Director > Resource Sizes and change a resource value. For example, you could increase the amount of RAM by 4 MB.

Creating Bound Missing VMs Times Out

This task happens immediately following package compilation, but before job assignment to agents. For example:

```
cloud_controller/0: Timed out pinging to f690db09-876c-475c-b65f-2ceca06a79 after 600 seconds (00:10:24)
```

This is most likely a NATS issue with the VM in question. To identify a NATS issue, inspect the agent log for the VM. Since the BOSH director is unable to reach the BOSH agent, you must access the VM using another method. You will likely also be unable to access the VM using TCP. In this case, access the VM using your virtualization console.

To diagnose:

1. Access the VM using your virtualization console and log in.
2. Navigate to the Credentials tab of the Pivotal Application Service (PAS) tile and locate the VM in question to find the VM credentials.
4. Run `cd /var/vcap/bosh/log`
5. Open the file `current`
6. First, determine whether the BOSH agent and director have successfully completed a handshake, represented in the logs as a “ping-pong”:

```
"reply_to":null,"director.f4b7d14-cb8f.19719501-000e-4f53-b755-58b6f36df58ab"}
```

This handshake must complete for the agent to receive instructions from the director.
7. If you do not see the handshake, look for another line near the beginning of the file, prefixed INFO: loaded new infrastructure settings . For example:
2013-10-03_14:35:21.83222 #[608] INFO: loaded new infrastructure settings:
{"vm"=>{"name"=>"vm-4d80ede4-b0a5-4992-aea6a0386e18e", "id"=>"vm-360"},
"agent_id"=>"56aea4ef-6aa9-4c39-8019-7024ccfdde4",
"networks"=>{"default"=>{"ip"=>"192.0.2.19",
"netmask"=>"255.255.255.0", "cloud_properties"=>{"name"=>"VMNetwork"},
"default"=>["dns", "gateway"],
"dns"=>["192.0.2.2", "192.0.2.17"], "gateway"=>"192.0.2.2",
"dns_record_name"=>"0.nats.default.cf-d729343071061.microbosh",
"mac"=>"00:50:56:9b:71:67"}}, "disks"=>{"system"=>0, "ephemeral"=>1,
"persistent"=>{}}, "ntp"=>[], "blobstore"=>{"provider"=>"dav",
"options"=>{"endpoint"=>"http://192.0.2.17:25250",
"user"=>"agent", "password"=>"agent"}},
"mbus"=>"nats://nats:nats@192.0.2.17:4222",
"env"=>{"bosh"=>{"password"=>"$6$40ftQ9K4rvvC/8ADZHW0"}}}

This is a JSON blob of key/value pairs representing the expected infrastructure for the BOSH agent. For this issue, the following section is the most
important:
"mbus"=>"nats://nats:nats@192.0.2.17:4222"

This key/value pair represents where the agent expects the NATS server to be. One diagnostic tactic is to try pinging this NATS IP address from the VM to
determine whether you are experiencing routing issues.

Install Exits With a Creates/Updates/Deletes App Failure or With a 403 Error
Scenario 1: Your PCF install exits with the following 403 error when you attempt to log in to the Apps Manager:
{"type": "step_finished", "id": "apps-manager.deploy"}
`fetch': 403 => Net::HTTPForbidden for https://login.api.example.net/oauth/authorizeresponse_type=code&client_id=portal&redirect_uri=https%3...
-- unhandled response (Mechanize::ResponseCodeError)

Scenario 2: Your PCF install exits with a creates/updates/deletes an app (FAILED -

error message with the following stack trace:

1)

1) App CRUD creates/updates/deletes an app
Failure/Error: Unable to find matching line from backtrace
CFoundry::TargetRefused:
Connection refused - connect(2)

In either of the above scenarios, ensure that you have correctly entered your domains in wildcard format:
1. Browse to the Operations Manager fully qualified domain name (FQDN).
2. Click the PAS tile.
3. Select HAProxy and click Generate Self-Signed RSA Certificate.
4. Enter your system and app domains in wildcard format, as well as optionally any custom domains, and click Save. Refer to PAS Cloud Controller for
explanations of these domain values.

© Copyright Pivotal Software Inc, 2013-2019

1542

2.5


### Install Fails When Gateway Instances Exceed Zero

If you configure the number of Gateway instances to be greater than zero for a given product, you create a dependency on PAS for that product installation. If you attempt to install a product tile with a PAS dependency before installing PAS, the install fails.

To change the number of Gateway instances, click the product tile, then select **Settings > Resource sizes > INSTANCES** and change the value next to the product Gateway job.

To remove the PAS dependency, change the value of this field to **0**.

### Out of Disk Space Error

PCF displays an **Out of Disk Space** error if log files expand to fill all available disk space. If this happens, rebooting the PCF installation VM clears the tmp directory of these log files and resolves the error.

If users receive **Out of Disk Space** errors when trying to push apps, this can mean that Diego cells may be running out of disk space capacity.

To perform a detailed analysis of disk usage by containers and host VMs in your PAS deployment, see **Examining GrootFS Disk Usage**.

### Installing BOSH Director Fails

If the DNS information for the PCF VM is incorrectly specified when deploying the PCF.ova file, installing BOSH Director fails at the “Installing Micro BOSH” step.

To resolve this issue, correct the DNS settings in the PCF Virtual Machine properties.

### Deleting Ops Manager Fails

Ops Manager displays an error message when it cannot delete your installation. This scenario might happen if the BOSH Director cannot access the VMs or is experiencing other issues. To manually delete your installation and all VMs, you must do the following:

1. Use your IaaS dashboard to manually delete the VMs for all installed products, with the exception of the Ops Manager VM.

2. SSH into your Ops Manager VM and remove the `installation.yml` file from `/var/tempest/workspaces/default/`.

   **Note:** Deleting the `installation.yml` file does not prevent you from reinstalling Ops Manager. For future deploys, Ops Manager regenerates this file when you click **Save** on any page in the BOSH Director.

Your installation is now deleted.

### Installing PAS Fails

© Copyright Pivotal Software Inc, 2013-2019

1543

2.5
The following sections describe errors that may occur when installing PAS and how to resolve them.

**Ops Manager Cannot Verify App Push**

If the DNS information for the PCF VM becomes incorrect after BOSH Director has been installed, installing PAS with Pivotal Operations Manager fails at the “Verifying app push” step.

To resolve this issue, correct the DNS settings in the PCF Virtual Machine properties.

**MySQL Monitor Not Running After Update**

When MySQL cannot communicate with UAA, Ops Manager shows the following error:

```
Error: 'mysql_monitor/12a3b456-cd7e-8fgh-9012-345678b90ijk (0)' is not running after update. Review logs for failed jobs: replication-canary.
```

If you see this error, create firewall rules that allow MySQL to reach UAA, using the [MySQL Network Communications](#) topic as a reference.

**Ops Manager Hangs During MicroBOSH Install or HAProxy States “IP Address Already Taken”**

During an Ops Manager installation, you might receive the following errors:

- The Ops Manager GUI shows that the installation stops at the “Setting MicroBOSH deployment manifest” task.
- When you set the IP address for the HAProxy, the “IP Address Already Taken” message appears.

When you install Ops Manager, you assign it an IP address. Ops Manager then takes the next two consecutive IP addresses, assigns the first to MicroBOSH, and reserves the second. For example:

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>203.0.113.1</td>
<td>Ops Manager (User assigned)</td>
</tr>
<tr>
<td>203.0.113.2</td>
<td>MicroBOSH (Ops Manager assigned)</td>
</tr>
<tr>
<td>203.0.113.3</td>
<td>Reserved (Ops Manager reserved)</td>
</tr>
</tbody>
</table>

To resolve this issue, ensure that the next two subsequent IP addresses from the manually assigned address are unassigned.

**Poor PCF Performance**

If you notice poor network performance by your PCF deployment and your deployment uses a Network Address Translation (NAT) gateway, your NAT gateway may be under-resourced.

**Troubleshoot**

To troubleshoot the issue, set a custom firewall rule in your IaaS console to route traffic originating from your private network directly to an S3-compatible object store. If you see decreased average latency and improved network performance, perform the solution below to scale up your NAT gateway.

**Scale Up Your NAT Gateway**

Perform the following steps to scale up your NAT gateway:

1. Navigate to your IaaS console.
2. Spin up a new NAT gateway of a larger VM size than your previous NAT gateway.
3. Change the routes to direct traffic through the new NAT gateway.
4. Spin down the old NAT gateway.

The specific procedures will vary depending on your IaaS. Consult your IaaS documentation for more information.
Common Issues Caused by Firewalls

This section describes various issues you might encounter when installing PAS in an environment that uses a strong firewall.

DNS Resolution Fails

When you install PCF in an environment that uses a strong firewall, the firewall might block DNS resolution. To resolve this issue, refer to the Troubleshooting DNS Resolution Issues section of the Preparing Your Firewall for Deploying PCF topic.
Advanced Troubleshooting with the BOSH CLI

This topic describes using the BOSH CLI to help diagnose and resolve issues with your Pivotal Cloud Foundry (PCF) deployment. Before using the information and techniques in this topic, review Diagnosing Problems in PCF.

To follow the steps in this topic, you must log in to the BOSH Director VM. The BOSH Director runs on the virtual machine (VM) that Ops Manager deploys on the first install of the BOSH Director tile.

After authenticating into the BOSH Director, you can run specific commands using the BOSH Command Line Interface (BOSH CLI). BOSH Director diagnostic commands have access to information about your entire Pivotal Cloud Foundry (PCF) installation.

Gather Credential and IP Address Information

Before you begin troubleshooting with the BOSH CLI, follow the instructions below to collect the information you need from the Ops Manager interface.

1. Open the Ops Manager interface by navigating to the Ops Manager fully qualified domain name (FQDN) in a web browser.

2. Click the BOSH Director tile and select the Status tab.

3. Record the IP address for the Director job. This is the IP address of the VM where the BOSH Director runs.

4. Select the Credentials tab.

5. Click Link to Credential to view the Director Credentials. Record these credentials.

6. Return to the Installation Dashboard.
7. *(Optional)* To prepare to troubleshoot the job VM for any other product, click the product tile and repeat the procedure above to record the IP address and VM credentials for that job VM.

8. Log out of Ops Manager.

    **Note:** Ensure that there are no Ops Manager installations or updates in progress while using the BOSH CLI.

---

### Log in to the Ops Manager VM with SSH

Use SSH to connect to the Ops Manager VM. Follow the instructions in one of the sections below to log in to the Ops Manager VM with SSH.

#### AWS

To log in to the Ops Manager VM with SSH in AWS, you need the key pair you used when you created the Ops Manager VM. To see the name of the key pair, click on the Ops Manager VM and locate the *key pair name* in the properties.

To log in to the Ops Manager VM with SSH in AWS, do the following:

1. Locate the Ops Manager FQDN on the AWS EC2 instances page.
2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive. For example:

   ```
   $ chmod 600 ops_mgr.pem
   ```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to log in to the Ops Manager VM with SSH. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

   ```
   $ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
   ```

#### Azure

To log in to the Ops Manager VM with SSH in Azure, you need the key pair you used when creating the Ops Manager VM. If you need to reset the SSH key, locate the Ops Manager VM in the Azure portal and click *Reset Password*.

To log in to the Ops Manager VM with SSH in Azure, do the following:

1. Locate the Ops Manager FQDN by selecting the VM in the Azure portal.
2. Run `chmod 600 ops_mgr.pem` to change the permissions on the `.pem` file to be more restrictive. For example:

   ```
   $ chmod 600 ops_mgr.pem
   ```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to log in to the Ops Manager VM with SSH. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

   ```
   $ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
   ```

#### GCP

To log in to the Ops Manager VM with SSH in GCP, do the following:

1. Confirm that you have installed the [Google Cloud SDK and CLI](https://cloud.google.com/sdk). See the [Google Cloud Platform documentation](https://cloud.google.com) for more information.
2. Initialize Google Cloud CLI, using a user account with Owner, Editor, or Viewer permissions to access the project. Ensure that the Google Cloud CLI can login to the project by running the command `gcloud auth login`.
3. From the GCP web console, navigate to *Compute Engine*.
4. Locate the Ops Manager VM in the VM Instances list.

5. Under Remote access, click the SSH dropdown and select View gcloud command.

6. Copy the SSH command that appears in the popup window.

7. Paste the command into your terminal window to SSH to the VM. For example:

   ```bash
   $ gcloud compute ssh "YOUR-VM" --zone "YOUR-ZONE-ID"
   ```

8. Run `sudo su ubuntu` to switch to the ubuntu user.

OpenStack

To log in to the Ops Manager VM with SSH in OpenStack, you need the key pair that you created in Step 2: Configure Security of Deploying Ops Manager to OpenStack. If you need to reset the SSH key, locate the Ops Manager VM in the OpenStack console and boot it in recovery mode to generate a new key pair.

To log in to the Ops Manager VM with SSH in OpenStack, do the following:

1. Locate the Ops Manager FQDN on the Access & Security page.

2. Run `chmod 600 ops_mgr.pem` to change the permissions on the .pem file to be more restrictive. For example:

   ```bash
   $ chmod 600 ops_mgr.pem
   ```

3. Run `ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN` to log in to the Ops Manager VM with SSH. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager. For example:

   ```bash
   $ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
   ```

vSphere

To log in to the Ops Manager VM with SSH in vSphere, you need the credentials used to import the PCF .ova or .ovf file into your virtualization system. You set these credentials when you installed Ops Manager.

To log in to the Ops Manager VM with SSH in vSphere, do the following:

1. From a command line, run `ssh ubuntu@OPS-MANAGER-FQDN` to log in to the Ops Manager VM with SSH. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.

2. When prompted, enter the password that you set during the .ova deployment into vCenter. For example:

   ```bash
   $ ssh ubuntu@my-opsmanager-fqdn.example.com
   Password: ***********
   ```

Log in to the BOSH Director VM

Follow the steps below to log in to the BOSH Director VM.

Create a Local BOSH Director Alias

1. Run the following command to create a local alias for the BOSH Director using the BOSH CLI:

   ```bash
   bosh alias-env MY-ENV -e DIRECTOR-IP-ADDRESS --ca-cert /var/tempest/workspaces/default/root_ca_certificate
   ```

   Replace the placeholder text with the following:

   * **MY-ENV**: Enter an alias for the BOSH Director, such as `gcp`.

© Copyright Pivotal Software Inc, 2013-2019
1. **Log in to the BOSH Director VM using one of the following options:**

   - **Internal User Store Login through UAA:** Log in to the BOSH Director VM using BOSH.
   - **External User Store Login through SAML:** Use an external user store to log in to the BOSH Director VM.

2. **Log in to the BOSH Director VM with UAA**

   1. Retrieve the Director password from the **BOSH Director > Credentials** tab. Alternatively, launch a browser and visit [https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/director_credentials](https://OPS-MANAGER-FQDN/api/v0/deployed/director/credentials/director_credentials) to obtain the password. Replace `OPS-MANAGER-FQDN` with the fully qualified domain name of Ops Manager.

   2. Run `bosh -e MY-ENV log-in` to log in to the BOSH Director VM. Replace `MY-ENV` with the alias for your BOSH Director. For example: `bosh -e gcp log-in`

   Follow the BOSH CLI prompts and enter the BOSH Director credentials to log in to the BOSH Director VM.

3. **Log in to the BOSH Director VM with SAML**

   1. Log in to your identity provider and use the following information to configure SAML Service Provider Properties:

      - **Service Provider Entity ID:** `bosh-uaa`
      - **ACS URL:** [https://DIRECTOR-IP-ADDRESS:8443/saml/SSO/alias/bosh-uaa](https://DIRECTOR-IP-ADDRESS:8443/saml/SSO/alias/bosh-uaa)
      - **Binding:** HTTP Post
      - **SLO URL:** [https://DIRECTOR-IP-ADDRESS:8443/saml/SSO/alias/bosh-uaa](https://DIRECTOR-IP-ADDRESS:8443/saml/SSO/alias/bosh-uaa)
      - **Binding:** HTTP Redirect
      - **Name ID:** Email Address

   2. Run `bosh -e MY-ENV log-in` to log in to the BOSH Director VM. Replace `MY-ENV` with the alias for your BOSH Director. For example: `bosh -e gcp log-in`

   Follow the BOSH CLI prompts and enter your SAML credentials to log in to the BOSH Director VM.

   ```
   Note: Your browser must be able to reach the BOSH Director in order to log in with SAML.
   ```

4. **Click Log in with organization credentials (SAML).**

5. **Copy the Temporary Authentication Code that appears in your browser.**

6. **You see a login confirmation.** For example:
Log in to the BOSH Director VM with SSH

Do the following steps to log in to the BOSH Director VM with SSH:

1. From Ops Manager, open the BOSH Director tile.

2. Select the Credentials tab.

3. Next to Bbr Ssh Credentials, click Link to Credential. A tab opens containing a JSON credential structure.

4. Copy the RSA PRIVATE KEY and paste it into a file named `bbr.pem`. Include `-----BEGIN RSA PRIVATE KEY-----` and `-----END RSA PRIVATE KEY-----`.

⚠️ warning: Pivotal recommends you keep the key secure. The key provides full access to the entire PCF environment.

5. Replace all `\n` characters in `bbr.pem` with a line break.

6. Copy `bbr.pem` to the `~/.ssh` directory on your machine.

7. Modify the permissions of the file by running `chmod 600 ~/.ssh/bbr.pem`.

8. Log in to the BOSH Director VM with SSH from your machine.

   ```bash
   ssh bbr@BOSH-DIRECTOR-IP -i ~/.ssh/bbr.pem
   ```

   ❗️ Note: If you are using GCP, ensure SSH port 22 is open for your BOSH Director VM in your GCP console. If the SSH port is not open, open it by creating a firewall rule.

9. Run `sudo -i` to get the root privilege.

Use the BOSH CLI for Troubleshooting

This section describes three BOSH CLI commands commonly used during troubleshooting.

- **VMs**: Lists the VMs in a deployment
- **Cloud Check**: Runs a cloud consistency check and interactive repair
- **SSH**: Starts an interactive session or executes commands with a VM

**BOSH VMs**

The `bosh vms` command provides an overview of the virtual machines that BOSH manages.

To use this command, run `bosh -e MY-ENV vms` to see an overview of all virtual machines managed by BOSH, or `bosh -e MY-ENV -d MY-DEPLOYMENT vms` to see only the virtual machines associated with a particular deployment. Replace `MY-ENV` with your environment, and, if using the `-d` flag, also replace `MY-DEPLOYMENT` with the name of a deployment.

When troubleshooting an issue with your deployment, `bosh vms` may show a VM in an unknown state. Run `bosh cloud-check` on a VM in an unknown state to instruct BOSH to diagnose problems with the VM.

You can also run `bosh vms` to identify VMs in your deployment, then use the `bosh ssh` command to log in to an identified VM with SSH for further troubleshooting.

`bosh vms` supports the following arguments:

- `--dns`: Report also includes the DNS A record for each VM
- `--vitals`: Report also includes load, CPU, memory usage, swap usage, system disk usage, ephemeral disk usage, and persistent disk usage for each
BOSH Cloud Check

Run the `bosh cloud-check` command to instruct BOSH to detect differences between the VM state database maintained by the BOSH Director and the actual state of the VMs. For each difference detected, `bosh cloud-check` can offer the following repair options:

- **Reboot VM**: Instructs BOSH to reboot a VM. Rebooting can resolve many transient errors.
- **Ignore problem**: Instructs BOSH to do nothing. You may want to ignore a problem in order to run `bosh ssh` and attempt troubleshooting directly on the machine.
- **Reassociate VM with corresponding instance**: Updates the BOSH Director state database. Use this option if you believe that the BOSH Director state database is in error and that a VM is correctly associated with a job.
- **Recreate VM using last known apply spec**: Instructs BOSH to destroy the server and recreate it from the deployment manifest that the installer provides. Use this option if a VM is corrupted.
- **Delete VM reference**: Instructs BOSH to delete a VM reference in the Director state database. If a VM reference exists in the state database, BOSH expects to find an agent running on the VM. Select this option only if you know that this reference is in error. Once you delete the VM reference, BOSH can no longer control the VM.

To use this command, run:

```
$ bosh -e MY-ENV -d MY-DEPLOYMENT cloud-check
```

Replace `MY-ENV` with your environment, and `MY-DEPLOYMENT` with your deployment.

Example Scenarios

**Unresponsive Agent**

```
$ bosh -e example-env -d example-deployment cloud-check
ccdb/0 (vm-3e37133c-bc33-450e-98b1-f86d5b63502a) is not responding:
- Ignore problem
- Reboot VM
- Recreate VM using last known apply spec
- Delete VM reference (DANGEROUS!)
```

**Missing VM**

```
$ bosh -e example-env -d example-deployment cloud-check
VM with cloud ID `vm-3e37133c-bc33-450e-98b1-f86d5b63502a` missing:
- Ignore problem
- Recreate VM using last known apply spec
- Delete VM reference (DANGEROUS!)
```

**Unbound Instance VM**

```
$ bosh -e example-env -d example-deployment cloud-check
VM `vm-3e37133c-bc33-450e-98b1-f86d5b63502a` reports itself as `ccdb/0` but does not have a bound instance:
- Ignore problem
- Delete VM (unless it has persistent disk)
- Reassociate VM with corresponding instance
```

**Out of Sync VM**

```
$ bosh -e example-env -d example-deployment cloud-check
VM `vm-3e37133c-bc33-450e-98b1-f86d5b63502a` is out of sync:
  expected `cfd7293407023a2c4206e1: ccdb/0` but got `cfd7293407023a2c4206e1: nats/0`:
- Ignore problem
- Delete VM (unless it has persistent disk)
```
BOSH SSH

Use `bosh ssh` to log in to the VMs in your deployment with SSH.

Follow the steps below to use `bosh ssh`:

1. Identify a VM to log in to with SSH. Run `bosh -e MY-ENV -d MY-DEPLOYMENT vms` to list the VMs in the given deployment. Replace `MY-ENV` with your environment alias and `MY-DEPLOYMENT` with the deployment name.

2. Run `bosh -e MY-ENV -d MY-DEPLOYMENT ssh VM-NAME/GUID`. For example:

   ```bash
   $ bosh -e example-env -d example-deployment ssh diego-cell/abcd0123-a012-b345-c678-9def01234567
   ```
Troubleshooting Slow Requests in Cloud Foundry

Page last updated:

This topic suggests ways that an operator of Cloud Foundry (CF) can diagnose the location of app request delays.

App Request Path

App requests typically transit the following components. Only the router (Gorouter) and app are within the scope of Cloud Foundry. Operators may have the HAProxy load balancer that comes with CF deployed, instead of or in addition to, an infrastructure load balancer.

You can use `time` to measure a request’s full round trip time from the client and back, examine `cf logs` output to measure the time just within Cloud Foundry, and add log messages to your app for fine-grained measurements of where the app itself takes time. By comparing these times, you can determine whether your delay comes from outside CF, inside the Gorouter, or in the app.

The following sections describe a scenario of diagnosing the source of delay for an app `app1`.

Measure Total Round-Trip App Requests

On a command line, run `time curl -v APP-ENDPOINT` to measure the total round-trip time for deployed app `app1`. For example:

```
5 time curl -v http://app1.app_domain.com
GET /hello HTTP/1.1
Host: app1.app_domain.com
User-Agent: curl/7.43.0
Accept: */*

HTTP/1.1 200 OK
Date: Tue, 14 Dec 2016 00:31:32 GMT
Server: nginx
X-Content-Type-Options: nosniff
X-Vcap-Request-Id: c30fad28-4972-46eb-7da6-9d07dc79b109
Content-Length: 602

hello world!
real 2m0.707s
user 0m0.005s
sys 0m0.007s
```

The `real` time output shows that the request to `http://app1.app_domain.com` took approximately 2 minutes, round-trip. This seems like an unreasonably long time, so it makes sense to find out where the delay is occurring. To narrow it down, the next step measures the part of that request response time that comes from within Cloud Foundry.

**Note:** If your `curl` outputs an error like `Could not resolve host: NONEXISTENT.com` then DNS failed to resolve. If `curl` returns normally but lacks a `X-Vcap-Request-Id`, the request from the Load Balancer did not reach Cloud Foundry.
Measure App Requests within Cloud Foundry

The `cf logs` command streams log messages from the Gorouter as well as from apps. To see the timestamps of Gorouter messages to and from your app, do the following:

1. If necessary, run `cf apps` to determine the name of the app.
2. Run `cf logs APP-NAME`. Replace `APP-NAME` with the name of the app.
3. From another terminal window, send a request to your app.
4. After your app returns a response, enter `Ctrl-C` to stop streaming `cf logs`.

For example:

```
$ cf logs app1
```

In the example above, the first line contains timestamps from the Gorouter for both when it received the request and what was its response time processing the request:

- `14/12/2016:00:31:32.348`: Gorouter receives request
- `response_time:120.00641734`: Gorouter round-trip processing time

This output shows that it took 120 seconds for the Gorouter to process the request, which means that the 2-minute delay above takes place within CF, either within the Gorouter or within the app.

To determine whether the app is responsible, add logging to your app to measure where it is spending time.

### Use App Logs to Locate Delays in CF

To gain a more detailed picture of where delays exist in your request path, augment the logging that your app generates. For example, call your logging library from the request handler to generate log lines when your app receives a request and finishes processing it:

```
2016-12-14T00:31:32.35-0800 [RTR/0] OUT app1.app_domain.com - [14/12/2016:00:31:32.348 +0000] "GET /hello HTTP/1.1" 200 0 0 "" HTTPClient/1.0 (2.7.1, nhttp 2.3.3 (2016-11-2)
```

Comparing the router access log messages from the previous section with the new app logs above, we can construct the following timeline:

- `14/12/2016:00:31:32.348`: Gorouter receives request
- `2016-12-14T00:32:32.35-0800 [APP/PROC/WEB/0]OUT app1 received request at [14/12/2016:00:32:32.348 +0000] with "vcap_request_id": "01144146-1c7a-4c77-77ab-49ac32866e9"
- `2016-12-14T00:32:32.35-0800 [APP/PROC/WEB/0]OUT app1 finished processing request at [14/12/2016:00:32:32.500 +0000] with "vcap_request_id": "01144146-1c7a-4c77-77ab-49ac32866e9"

The timeline indicates that the Gorouter took close to 60 seconds to send the request to the app and another 60 seconds to receive the response from the app. This suggests a delay either with the Gorouter, or in network latency between the Gorouter and Diego cells hosting the app.

### Time the Gorouter Processing

To determine whether a Gorouter delay comes from the Gorouter itself or network latency between the Gorouter and the app, log into the Gorouter and compare the response times from calling the app two different ways:

- Call the app through the router proxy to process requests through the Gorouter
- Directly call a specific instance of the app, bypassing Gorouter processing

© Copyright Pivotal Software Inc, 2013-2019
To make this comparison, do the following:

1. Log in to the router using `bosh ssh`. See the [BOSH SSH documentation](#) for more information.

2. Use `time` and `curl` to measure the response time of an app request originating from and processing through the Gorouter, but not running through the client network or load balancer:

   ```
   ```

3. Obtain the IP address and port of a specific app instance by running the following and recording associated `host` and `port` values:

   ```
   $ cf curl/v2/apps/$(cf app app1 --guid)/stats
   
   { 
     "v": 
     "status": "RUNNING", 
     "stats": 
     [ 
       "host": "10.10.148.39", 
       "port": 60052, 
       [ ] 
     ], 
     [ ] 
   }
   ```

4. Use the IP address and port values to measure response time calling the app instance directly, bypassing Gorouter processing:

   ```
   $ time curl http://APP-HOST-IP:APP-PORT
   ```

If the Gorouter and direct response times are similar, it suggests network latency between the Gorouter and the Diego cell. If the Gorouter time is much longer than the direct-to-instance time, the Gorouter is slow processing requests. The next section explains why the Gorouter might be slow.

### Potential Causes for Gorouter Latency

- Routers are under heavy load from incoming client requests.
- Apps are taking a long time to process requests. This increases the number of concurrent threads held open by the Gorouter, reducing capacity to handle requests for other apps.

### Operations Recommendations

- Monitor CPU load for Gorouters. At high CPU (70%+), latency increases. If the Gorouter CPU reaches this threshold, consider adding another Gorouter instance.
- Monitor latency of all routers using metrics from the Firehose. Do not monitor the average latency across all routers. Instead, monitor them individually on the same graph.
- Consider using [Pingdom](#) against an app on your Cloud Foundry deployment to monitor latency and uptime.
- Consider enabling access logs on your load balancer. See your load balancer documentation for how. Just as we used Gorouter access log messages above to determine latency from the Gorouter, you can compare load balancer logs to identify latency between the load balancer and the Gorouter. You can also compare load balancer response times with the client response times to identify latency between client and load balancer.
- Deploy a nozzle to the Loggregator Firehose to track metrics for the Gorouter. Available metrics include:
  - CPU utilization
  - Latency
  - Requests per second
Troubleshooting TCP Routes

Rule Out the App

If you are having TCP routing issues with an app, follow the procedure below to determine what to troubleshoot.

Note: If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This disables TCP routing.

Prerequisites

This procedure requires that you have the following:

- An app with TCP routing issues.
- A TCP domain. See Routes and Domains for more information about creating a TCP domain.
- A simple HTTP web app that you can use to curl.

Procedure

1. Push a simple HTTP app using your TCP domain by entering the following command:

   $ cf push MY-APP -d tcp.MY-DOMAIN --random-route

2. Curl your app on the port generated for the route. For example, if the port is 1024:

   $ curl tcp.MY-DOMAIN:1024

3. If the curl request fails to reach the app, proceed to the next section: Rule Out DNS and the Load Balancer.

4. If the curl request to your simple app succeeds, curl the app you are having issues with.

5. If you cannot successfully curl your problem app, TCP routing is working correctly. There is an issue with the app you cannot successfully curl.

Rule Out DNS and the Load Balancer

1. Curl the TCP router healthcheck endpoint:

   $ curl tcp.MY-DOMAIN:80/health -v

2. If you receive a 200 OK response, proceed to the next section: Rule Out the Routing Subsystem.

3. If you do not receive a 200 OK, your load balancer may not be configured to pass through the healthcheck port. Continue following this procedure to test your load balancer configuration.

4. Confirm that your TCP domain name resolves to your load balancer:

   $ dig tcp.MY-DOMAIN

   tcp.MY-DOMAIN. 300 IN A 123.456.789.123

5. As an admin user, list the reservable ports configured for the default-tcp router group:

© Copyright Pivotal Software Inc, 2013-2019
6. Choose a port from the `reservable_ports` range and curl the TCP domain on this port. For example, if you chose port 1024:

```bash
curl tcp.MY-DOMAIN:1024 -v
```

7. If you receive an immediate rejection, then the TCP router likely rejected the request because there is no route for this port.

8. If your connection times out, then you need to configure your load balancer to route all ports in `reservable_ports` to the TCP routers.

## Rule Out the Routing Subsystem

Send a direct request to the TCP router to confirm that it routes to your app.

1. SSH into your TCP router.

2. Curl the port of your route using the IP address of the router itself. For example, if the port reserved for your route is `1024`, and the IP address of the TCP router is `10.0.16.18`:

```bash
curl 10.0.16.18:1024
```

3. If the curl is successful, then the load balancer either:
   - cannot reach the TCP routers, or
   - is not configured to route requests to the TCP routers from the `reservable_ports`

4. If you cannot reach the app by curling the TCP router directly, perform the following steps to confirm that your TCP route is in the routing table.
   a. Record the `Tcp Emitter Credentials` from the UAA row in the PAS Credentials tab. This OAuth client has permissions to read the routing table.
   b. Install the UAA CLI `uaac`:

```bash
gem install cf-uaac
```

c. Obtain a token for this OAuth client from UAA by providing the client secret:

```bash
```

d. Obtain an access token:

```bash
```

5. In this output, each route mapping has the following:
   - **port**: your route port
   - **backend_ip**: an app instance mapped to the route
   - **backend_port**: the port number for the app instance mapped to the route

g. If your route port is not in the response, then the `tcp_emitter` may be unable to register TCP routes with the routing API. Look at the logs for `tcp_emitter` to see if there are any errors or failures.
h. If the route is in the response, but you were not able to curl the port on the TCP route directly, then the TCP router may be unable to reach the routing API. Look at the logs for tcp_router to see if there are any errors or failures.
Troubleshooting Router Error Responses

Page last updated:

This topic helps operators to better understand if 502's are a result of the Pivotal Application Service (PAS) Platform or an application.

Points of Failure

There are different points of failure in which 502's can come from:

1. Infrastructure
   - Load Balancer
   - Network

2. Platform - PCF
   - Gorouter
   - Diego Cell(s)

3. Application

In the **Infrastructure**, 502’s can occur in the following way:

- From the Load Balancer, 502's can surface when the Gorouters are not receiving traffic at all.
- This can be observed if the Load Balancer is logging 502's but the Gorouters are not.

In the **Platform**, 502's can occur in the following ways:

- If the Gorouter is unable to connect to the application container:
  - TCP dial issues (can’t make an initial connection to the backend). The Gorouter will retry TCP dial errors up to three times, if it still fails then a 502 will be returned to the client and logged to the access.log. This may be due to:
    - An application that is unresponsive (which indicates an issue with the application)
    - The Gorouter has a stale route (which indicates an issue with the platform)
    - The application container is corrupted (which indicates a problem with the platform)
    - These types of errors may look like this within the gorouter.log:

```
```

- If the Gorouter successfully dials the endpoint but an error occurs:
  - read: connection reset by peer errors can occur when the application closes the connection abruptly with a TCP RST packet and not the expected FIN-ACK. This will cause the Gorouter to retry the next endpoint. Note, Gorouter does not currently retry on write: connection reset by peer failures.
  - TLS Handshake errors. When these errors occur, the Gorouter will retry up to three times and if it’s still failing then a 502 may be returned. These errors appear similar to the following in the gorouter.log (and a 502 will be logged in the access.log):

```
```

- If the Gorouter successfully connects to the endpoint, but an error occurs while the request is in transport (i.e. Gorouter has not received a response from the endpoint):
  - Prior to PCF 2.0, there was a bug that logged a 502 for requests canceled by clients before the server responded with headers. PCF 2.0 and beyond, if the same situation occurs, a 499 is returned.

In an **Application**, 502’s can occur in the following ways (Note: the Gorouter will not retry any error response that is returned by the application):

- If 502’s are only occurring from a particular application instance(s) and not all of the applications on the platform, then it is likely an application-related error (i.e. application is overloaded, unresponsive, can’t connect to database, etc.).
• If all applications are experiencing 502’s, then it could either be a platform issue (possible misconfiguration) or an application issue (i.e. all applications are unable to connect to an upstream database).

General Debugging Steps

Here are general debugging steps for any issue resulting with 502 error codes:

• Gather the Gorouter logs & Diego Cell logs at the time of the incident. To SSH into the router VM, see Advanced Troubleshooting with the BOSH CLI. To download the router VM logs from Ops Manager, see Monitoring PCF VMs from Ops Manager.

• Review the logs and consider the following questions:
  1. Which errors are the Gorouters returning?
  2. Is the Gorouter’s routing table accurate (are the endpoints for the route as expected)?
  3. Do the Diego Cell logs have anything interesting about unexpected app crashes and/or restarts?
  4. Is the application healthy and handling requests successfully? (try using request tracing headers to verify)

• Was there a recent platform change/upgrade that caused an increase in 502’s?

• Are there any suspicious metrics spiking? How is CPU & Memory utilization?

Gorouter Error Classification Table

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Status Code</th>
<th>Source of Issue</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dial</td>
<td>502</td>
<td>Application or Platform</td>
<td>logs with error <a href="#">dial tcp</a></td>
</tr>
</tbody>
</table>
| AttemptedTLSWithNonTLSBackend | 525      | Platform*                | - logs with error [tls: first record does not look like a TLS handshake](#)
|                         |             |                          | - backend_tls_handshake_failed metric increments                          |
| HostnameMismatch        | 503         | Platform                 | - logs with error [x509: certificate is valid for <x> not <y>](#)
|                         |             |                          | - backend_invalid_id metric increments                                    |
| UntrustedCert           | 526         | Platform                 | - logs with an error prefix [x509: certificate signed by unknown authority](#)
|                         |             |                          | - backend_invalid_tls_cert metric increments                              |
| RemoteFailedCertCheck   | 496         | Platform                 | logs with error remote error: tls: bad certificate                      |
| ContextCancelled        | 499         | Client/App               | logs with error context canceled                                         |
| RemoteHandshakeFailure  | 525         | Platform                 | - logs with error remote error: tls: handshake failure                   |
|                         |             |                          | - backend_tls_handshake_failed metric increments                          |

*Note: any platform issue could be the result of a misconfiguration

For each of the above errors, there will be a [backend-endpoint-failure](#) log line in gorouter.log and an error message in gorouter.err.log. Additionally, the access.log will record the request status codes.
Troubleshooting Ops Manager for VMware vSphere

This guide provides help with diagnosing and resolving issues that are specific to Pivotal Cloud Foundry (PCF) deployments on VMware vSphere.

For infrastructure-agnostic troubleshooting help, refer to Diagnosing Problems in PCF.

Common Issues

The following sections list common issues you might encounter and possible resolutions.

PCF Installation Fails

If you modify the vCenter Statistics Interval Duration setting from its default setting of 5 minutes, the PCF installation might fail at the MicroBOSH deployment stage, and the logs might contain the following error message: 

```
The specified parameter is not correct, interval.
```

This failure happens because Ops Manager expects a default value of 5 minutes, and the call to this method fails when the retrieved value does not match the expected default value.

To resolve this issue, launch vCenter, navigate to Administration > vCenter Server Settings > Statistics, and reset the vCenter Statistics Interval Duration setting to 5 minutes.

BOSH Automated Installation Fails

Before starting a Pivotal Application Service (PAS) deployment, you must set up and configure a vSphere cluster.

If you enable vSphere DRS (Distributed Resource Scheduler) for the cluster, you must set the Automation level to Partially automated or Fully automated.

If you set the Automation level to Manual, the BOSH automated installation will fail with a `power_on_vm` error when BOSH attempts to create virtual VMs.

Ops Manager Loses Its IP Address After HA or Reboot
Ops Manager can lose its IP address and use DHCP due to an issue in the open source version of VMware Tools. For more information to troubleshoot this issue, see IP is not Getting Assigned in vSphere after Ops Manager Restart in the Pivotal Support Knowledge Base.

**Cannot Connect to the OVF in a Browser**

If you deployed the OVF file but cannot connect to in a browser, check that the network settings you entered in the wizard are correct.

1. Access the PCF installation VM using the vSphere Console. If your network settings are configured incorrectly, you cannot SSH into the installation VM.
2. Log in using the credentials you provided when you imported the PCF .ova in vCenter.
3. Confirm that the network settings are correct by checking that the ADDRESS, NETMASK, GATEWAY, and DNS-NAMESERVERS entries are correct in `/etc/network/interfaces`
4. If any of the settings are incorrect, run `sudo vi /etc/network/interfaces` and correct the incorrect entries.
5. In vSphere, navigate to the Summary tab for the VM and confirm that the network name is correct.

![VM with Network Interface Configuration](image)

6. If the network name is incorrect, right-click on the VM, select Edit Settings > Network adapter 1, and select the correct network.
7. Reboot the installation VM.

**Installation Fails with Failed Network Connection**

If you experience a communication error while installing Ops Manager or MicroBOSH Director, check the following settings.

- Ensure that the routes are not blocked. vSphere environments use NSX for firewall and NAT/SNAT translation and load balancing. All communication between PCF VMs and vCenter or ESXi hosts route through the NSX firewall and are blocked by default.
- Open port 443. Ops Manager and MicroBOSH Director VMs require access to vCenter and all ESX through port 443.
- Allocate more IP addresses. BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP address range during installation. BOSH uses these IP addresses during installation to compile and deploy VMs, install PAS, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and PAS.

**Insufficient External Database Permissions**

Upgrade issues can be caused when the external database user used for the network policy DB is given insufficient permissions. To avoid this upgrade issue, ensure that the networkpolicyserver database user has the `ALL PRIVILEGES` permission.