

Table of Contents

Table of Contents	1
MySQL for PCF	2
Release Notes	6
Back Up MySQL for Pivotal Cloud Foundry	8

MySQL for PCF

This is documentation for the MySQL service for [Pivotal Cloud Foundry](#) (PCF).

Release Notes

Consult the [Release Notes](#) for information about changes between versions of this product.

Limitations

- Only the InnoDB storage engine is supported; it is the default storage engine for new tables. Attempted use of other storage engines (including MyISAM) may result in data loss.
- All databases are managed by shared, multi-tenant server processes. Although data is securely isolated between tenants using unique credentials, application performance may be impacted by noisy neighbors.
- Round-trip latency between database nodes must be less than five seconds; if the latency is higher than this nodes will become partitioned. If more than half of cluster nodes are partitioned the cluster will lose quorum and become unusable until manually bootstrapped.
- [MariaDB Galera Cluster - Known Limitations](#).

Known Issues

- If a cluster node becomes a member of a partitioned minority of cluster nodes, proxy instances will route new connections to a healthy cluster node. Existing client connections will see writes rejected but connections will not be severed, and may have to wait for a timeout before they can reconnect.
- All proxy instances use the same method to determine cluster health but during brief node failures proxy instances may come to different conclusions. Determination of which cluster node should be considered primary is not synchronized across proxy instances. If proxy instances diverge in their view of which node is primary, connections through multiple proxy instances may reach different cluster nodes. This is only an issue for tables that receive highly concurrent writes. In this scenario, multiple clients writing to the same table can obtain locks on the same row, resulting in a deadlock; one commit will succeed, all others will fail and must be retried. This can be prevented by configuring your load balancer to route connections to only one proxy instance at a time.
- Once the product is deployed with operator-configured proxy IPs, the number of proxy instances can not be reduced, nor can the configured IPs be removed from the **Proxy IPs** field. If instead the product is initially deployed without proxy IPs, IPs added to the **Proxy IPs** field will only be used when adding additional proxy instances, scaling down is unpredictably permitted, and the first proxy instance can never be assigned an operator-configured IP.

Installation

This product requires Pivotal Cloud Foundry version 1.3.4 or greater.

1. Download the product file from [Pivotal Network](#).
2. Upload the product file to your Ops Manager installation.
3. Click **Add** next to the uploaded product description in the Available Products view to add this product to your staging area.
4. Click the newly added tile to review configurable [Settings](#).
5. Click **Apply Changes** to deploy the service.

Settings

Service Plan

A single service plan enforces quotas of 100MB of storage per database and 40 concurrent connections per user by

default. Users of Operations Manager can configure these plan quotas. Changes to quotas will apply to all existing database instances as well as new instances. In calculating storage utilization, indexes are included along with raw tabular data.

The name of the plan is **100mb-dev** by default and is automatically updated if the storage quota is modified.

Provisioning a service instance from this plan creates a MySQL database on a multi-tenant server, suitable for development workloads. Binding applications to the instance creates unique credentials for each application to access the database.

Note: The service plan quota enforcer will remove your applications permissions if you go over your limit. This can be remedied by updating to a larger plan or lowering your instance storage usage.

Proxy

The proxy tier is responsible for routing connections from applications to healthy MariaDB cluster nodes, even in the event of node failure.

Applications are provided with a hostname or IP address to reach a database managed by the service (for more information, see [About Service Binding](#)). By default, the MySQL service will provide bound applications with the IP of the first instance in the proxy tier. Even if additional proxy instances are deployed, client connections will not be routed through them. This means the first proxy instance is a single point of failure.

In order to eliminate the first proxy instance as a single point of failure, operators must configure a load balancer to route client connections to all proxy IPs, and configure the MySQL service to give bound applications a hostname or IP address that resolves to the load balancer.

Configuring a load balancer

In older versions of the product, applications were given the IP of the single MySQL server in bind credentials. When upgrading to v1.4, existing applications will continue to function but in order to take advantage of high availability features they must be rebound to receive either the IP of the first proxy instance or the IP/hostname of a load balancer.

In order to configure a load balancer with the IPs of the proxy tier before v1.4.0 is deployed and prevent applications from obtaining the IP of the first proxy instance, the product enables an operator to configure the IPs that will be assigned to proxy instances. The following instructions applies to the **Proxy** settings page for the MySQL product in Operation Manager.

- In the **Proxy IPs** field, enter a list of IP addresses that should be assigned to the proxy instances. These IPs must be in the CIDR range configured in the Director tile and not be currently allocated to another VM. Look at the **Status** pages of other tiles to see what IPs are in use.
- In the **Binding Credentials Hostname** field, enter the hostname or IP that should be given to bound applications for connecting to databases managed by the service. This hostname or IP should resolve to your load balancer and be considered long-lived. When this field is modified applications must be rebound to receive updated credentials.

Configure your load balancer to route connections for a hostname or IP to the proxy IPs. As proxy instances are not synchronized, we recommend configuring your load balancer to send all traffic to one proxy instance at a time until it fails, then failover to another proxy instance. For details, see [Known Issues](#).

Important: To configure your load balancer with a healthcheck or monitor, use TCP against port **1936**. Unauthenticated healthchecks against port 3306 will cause the service to become unavailable, and will require manual intervention to fix.

Adding a load balancer after an initial deploy

If v1.4.0 is initially deployed without a load balancer and without proxy IPs configured, a load balancer can be setup later to remove the proxy as a single point of failure. However, there are several implications to consider:

- Applications will have to be rebound to receive the hostname or IP that resolves to the load balancer. To rebound: unbind your application from the service instance, bind it again, then restage your application. For more information see [Managing Service Instances with the CLI](#). In order to avoid unnecessary rebounding, we recommend configuring a load balancer before deploying v1.4.0.
- Instead of configuring the proxy IPs in Operations manager, use the IPs that were dynamically assigned by looking at the **Status** page. Configuration of proxy IPs after the product is deployed with dynamically assigned IPs is not well supported; see [Known Issues](#).

Lifecycle Errands

Two lifecycle errands are run by default: the **broker registrar** and the **smoke test**. The broker registrar errand registers the broker with the Cloud Controller and makes the service plan public. The smoke test errand runs basic tests to validate that service instances can be created and deleted, and that applications pushed to Elastic Runtime can be bound and write to MySQL service instances. Both errands can be turned on or off on the **Lifecycle Errands** page under the **Settings** tab.

Resource Config

Instance Capacity

An operator can configure how many database instances can be provisioned (instance capacity) by configuring the amount of persistent disk allocated to the MySQL server nodes. The broker will provision a requested database if there is sufficient unreserved persistent disk. This can be managed using the Persistent Disk field for the MySQL Server job in the Resource Config setting page in Operations Manager. Not all persistent disk will be available for instance capacity; about 2-3 GB is reserved for service operation. Adding nodes to the cluster increases durability, not capacity. Multiple backend clusters, to increase capacity or for isolation, are not yet supported.

In determining how much persistent disk to make available for databases, operators should also consider that MariaDB servers require sufficient CPU, RAM, and IOPS to promptly respond to client requests for all databases.

Provisioning and Binding via Cloud Foundry

As part of installation the product is automatically registered with [Pivotal Cloud Foundry](#) Elastic Runtime (see [Lifecycle Errands](#)). On successful installation, the MySQL service is available to application developers in the Services Marketplace, via the web-based Developer Console or `cf marketplace`. Developers can then provision instances of the service and bind them to their applications:

```
$ cf create-service p-mysql 100mb-dev mydb
$ cf bind-service myapp mydb
$ cf restart myapp
```

For more information on use of services, see the [Services Overview](#).

Example Application

To help application developers get started with MySQL for PCF, we have provided an example application, which can be [downloaded here](#). Instructions can be found in the included README.

Service Dashboard

Cloud Foundry users can access a service dashboard for each database from Developer Console via SSO. The dashboard displays current storage utilization and plan quota. On the Space page in Developer Console, users with the SpaceDeveloper role will find a **Manage** link next to the instance. Clicking this link will log users into the service dashboard via SSO.

HAProxy Statistics Dashboard

The service provides a dashboard where administrators can observe metrics for each instance in the proxy tier. Metrics include the number of clients currently connected and the number of connections made to each of the backend database nodes.

This statistics dashboard for each proxy instance can be found at: `http://haproxy-<job index>.p-mysql.<system-domain>`. Job index starts at 0; if you have two proxy instances deployed and your system-domain is `example.com`, dashboards would be accessible at `http://haproxy-0.p-mysql.example.com` and `http://haproxy-1.p-mysql.example.com`.

Basic auth credentials are required to access the dashboard. These can be found in the Credentials tab of the MySQL product in Operations Manager.

Back Ups

See [Back Up MySQL for PCF](#).

Note: For information about backing up your PCF installation, refer to the [Backing Up Pivotal Cloud Foundry](#) topic.

Release Notes

1.5.0

- **AWS support:** The clustered database service can now be deployed on Amazon Web Services from the Operations Manager Web UI.
 - Deployment is limited to a single Availability Zone. Look for multi-AZ in future releases.
 - Single availability zone is a limitation on AWS. Operations Manager on vSphere continues to support deployment to multiple availability zones.
 - The default instance type for the cluster nodes on AWS is m3.large.
 - All jobs are deployed with SSD for ephemeral and persistent disk.
- **IaaS agnostic**
 - The same product can be deployed to both AWS and vSphere
 - Precompiled packages are no longer included
 - p-mysql 1.5.0 requires Operations Manager 1.4.0
- **New proxy tier**
 - Improved availability: We have entirely re-written the proxy to eliminate situations where clients could hang when a cluster node was unhealthy.
 - A dashboard that clearly displays node health in real time
- **Upgrade support:** This product can be automatically upgraded from version 1.3.2 or 1.4.0
- **Cluster node resources increased for vSphere:** The default resources are now 4GB RAM, 2 CPU, 10GB persistent disk
- **Faster compilation:** Default resource for the compilation jobs on vSphere are now 4GB RAM, 4 CPU, 20GB persistent disk
- **Bug fix:** Fix broker-deregistrar errand to succeed even when MySQL service is broken
- **Bug fix:** Quota enforcer could fail when broker hasn't finished initializing

1.4.0

- **High Availability:** database server is now clustered and synchronously replicated using MariaDB Galera Cluster. A copy of each database resides on all cluster nodes, and writes to any database are replicated to all copies. All client connections are routed to a primary cluster node, and in the event of a node failure the proxy tier manages failover, routing client connections to a healthy cluster node. MySQL server, proxy, and broker jobs can all be scaled out horizontally for increased availability, eliminating single points of failure.
- **Improved logging and monitoring:** route-registration on the broker is now an independent process
- **Bug fix:** calculation of storage utilization for the purposes of quota enforcement when multiple apps are bound
- **Bug fix:** format of jdbcUrl connection string (found in VCAP_SERVICES on bind)

Notes on High Availability

- When upgrading from an older version, applications must be rebound to take advantage of high availability features. To rebound: unbind your application from the service instance, bind it again, then restage your application. For more information see [Managing Service Instances with the CLI](#).
- Elimination of the proxy as a single point of failure requires configuration of an external load balancer to route connections to proxy instances. For details, see [Proxy Settings](#).
- See [Known Issues](#).

1.3.2

- **Updated stemcell addresses bash-shellshock vulnerabilities:** resolves CVEs discussed [here](#) and [here](#).

1.3.0

- **Syslog forwarding:** Syslogs are now streamed to the same host and port configured in Elastic Runtime settings
- **Dynamic instance capacity management :** Previously operators had to manually configure the maximum number of service instances permitted by the server. This required manual calculation and a knowledge of required system headroom. Admins can now manage instance capacity simply by adjusting persistent disk allocated to mysql nodes. Remaining instance capacity is determined dynamically by subtracting a safe estimate for system headroom and reserved storage for provisioned instances.
- **Trusty stemcell:** Server and broker are now deployed on Ubuntu “Trusty” 14.04 LTS stemcells, providing improved security, performance, and a smaller resource footprint.
- **Least necessary privileges:** The MySQL service dashboard uses a new, limited permission OAuth scope to determine whether a user currently has access to a service instance. The dashboard no longer has full read access to a user’s account.
- **Precompiled packages:** Most packages have been precompiled for the targeted stemcell. This will lower initial deployment times, at the cost of a larger download.

1.2.0

- Product renamed to ‘MySQL for Pivotal CF’
- Plan attributes are configurable: max storage per database, max concurrent connections per user, and max databases
- Plan name is determined dynamically based on configured storage quota
- Plan features include disclaimer that the service is not for production use
- Developers can SSO to a service dashboard that displays storage utilization
- Security fixes including updates to Rails
- Service broker is registered by URL (rather than by IP). Typically has the format `https://p-mysql.<cf-domain>`.
- Lifecycle errands are used to register the broker and run tests that verify the deployment.
- Improved logging in service broker

The following components will be re-deployed:

- cf-mysql-broker
- mysql

New components:

- broker-registrar
- broker-deregistrar
- acceptance-tests

1.1.0

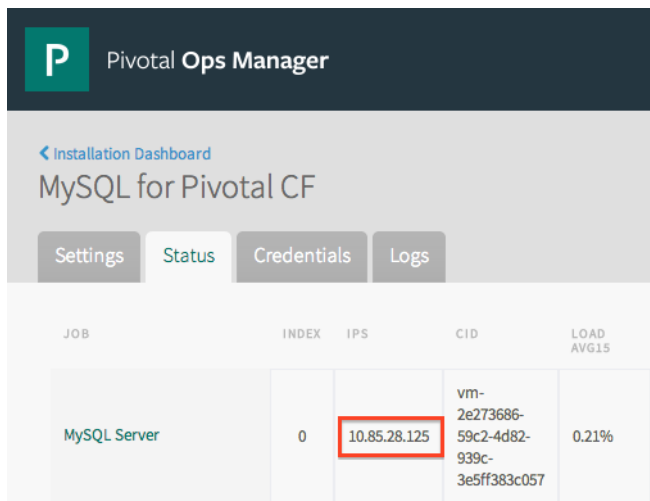
- Updated the format of metadata fields in the broker catalog endpoint and added additional fields. For more information, see Catalog Metadata.
- Updated Ruby to version 2.0.0p353 to fix a vulnerability in 1.9.3p448.
- Requests to delete a service instance or binding now get a 200 response with an empty JSON body instead of a 204.
- The broker now returns a clear error when there is no more capacity for additional instances during a provision request. The response has status code `507`. The user-facing error message is “Service plan capacity has been reached.”

The following components will be re-deployed:

- cf-mysql-broker
- mysql

Back Up MySQL for Pivotal Cloud Foundry

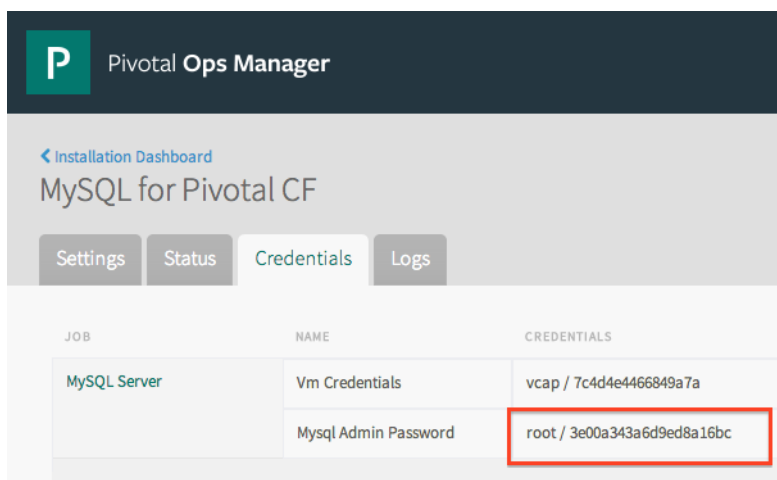
1. Locate the IP address for the MySQL node in the Status tab.



The screenshot shows the Pivotal Ops Manager interface for MySQL for Pivotal CF. The 'Status' tab is selected. A table lists the MySQL Server job with columns for JOB, INDEX, IPS, CID, and LOAD AVG15. The IP address 10.85.28.125 is highlighted in a red box.

JOB	INDEX	IPS	CID	LOAD AVG15
MySQL Server	0	10.85.28.125	vm-2e273686-59c2-4d82-939c-3e5ff383c057	0.21%

- Locate the root password for the MySQL server in the Credentials tab.



The screenshot shows the Pivotal Ops Manager interface for MySQL for Pivotal CF. The 'Credentials' tab is selected. A table lists the MySQL Server job with columns for JOB, NAME, and CREDENTIALS. The root password is highlighted in a red box.

JOB	NAME	CREDENTIALS
MySQL Server	Vm Credentials	vcap / 7c4d4e4466849a7a
	Mysql Admin Password	root / 3e00a343a6d9ed8a16bc

- Dump the data from the server and save it:

```
$ mysqldump -u root -p -h 172.16.78.60 --all-databases > user_databases.sql
```