


Table of Contents

Table of Contents	1
MySQL for Pivotal Cloud Foundry	2
Release Notes	13
Known Issues	18
Frequently Asked Questions	21
Cluster Scaling, Node Failure, and Quorum	22
Cluster Configuration	25
Proxy for MySQL for Pivotal Cloud Foundry	27
Creating Application Security Groups for MySQL	30
Monitoring the MySQL Service	31
Determining Cluster State	35
Bootstrapping a Galera Cluster	36
Backing Up MySQL for Pivotal Cloud Foundry	42
Scaling Down MySQL	50
Rotating MySQL for PCF Credentials	52
Running mysql-diag	55

MySQL for Pivotal Cloud Foundry

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

 **MySQL for PCF 1.8 is no longer supported.** The support period for version 1.8 has expired. To stay up to date with the latest software and security updates, please plan to upgrade to MySQL for PCF version 1.9 or greater.

Product Snapshot

The following table provides version and version-support information about MySQL for PCF:

Element	Details
Version	v1.8.11
Release date	August 11, 2017
Software component version	MariaDB v10.1.18, Galera v25.3.18
Compatible Ops Manager version(s)	v1.8.x, v1.9.x, v1.10.x
Compatible Elastic Runtime version(s)	v1.8.x, v1.9.x, v1.10.x
IaaS support	AWS, Azure, OpenStack, and vSphere
IPsec support	Yes

Upgrading to the Latest Version

Consider the following compatibility information before upgrading MySQL.

For more information, refer to the full [Product Compatibility Matrix](#).

Ops Manager Version	Supported Upgrades from Imported MySQL Installation	
	From	To
v1.8.x, v1.9.x, v1.10.x	v1.6.1 – v1.6.25	Next v1.6.x release – v1.6.26
		v1.7.0 – v1.7.32
		v1.8.0 - v1.8.11
	v1.7.0 – v1.7.31	Next v1.7.x release – v1.7.32
		v1.8.0 - v1.8.11
	v1.8.0 – v1.8.10	Next v1.8.x release - v1.8.11

Release Notes

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

Overview

The MySQL for PCF product delivers a fully managed, “Database as a Service” to Cloud Foundry users. When installed, the tile deploys and maintains a single or three-node cluster running a recent release of [MariaDB](#), SQL Proxies for super-fast failover, and Service Brokers for Cloud Foundry integration. We work hard to ship the service configured with sane defaults, following the principle of least surprise for a general-use relational database service.

When installed, Developers can attach a database to their applications in as little as two commands, `cf create-service` and `cf bind-service`.

Connection credentials are automatically provided in the [standard manner](#). Developers can select from a menu of service plans options, which are configured by the platform operator.

Two configurations are supported:


	Single	Highly Available
MySQL	1 node	3-node cluster
SQL Proxy	1 node	2 nodes
Service Broker	1 node	2 nodes
High Availability	-	Yes
Multi-AZ Support	-	Yes *
Rolling Upgrades	-	Yes
Automated Backups	Yes	Yes
Customizable Plans	Yes	Yes
Customizable VM Instances	Yes	Yes
Plan Migrations	Yes	Yes
Encrypted Communication	Yes †	Yes †
Encrypted Data at-rest	-	-
Long-lived Canaries	-	Replication Canary

(*) vSphere and AWS (v1.8.0-edge.15 and later)

(†) Requires IPSEC BOSH plug-in

Limitations

When deployed in HA configuration, MySQL for PCF is deployed using multiple master nodes. In this configuration, there are some changes to be aware of that make accessing the service different than a traditional single MySQL database server.

- Single and three-node clusters are the only supported topologies. Ops Manager will allow the Operator to set the number of instances to other values, only one and three are advised. Please see the [note](#) in the Cluster Behavior document.
- Although two Proxy instances are deployed by default, there is no automation to direct clients from one to the other. To address this, configure a load balancer as described in the [Proxy](#) section.
- Only the InnoDB storage engine is supported; it is the default storage engine for new tables. 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.
- See also the list of [Known Limitations](#)  in MariaDB cluster.

Known Issues

Consult the [Known Issues](#) topic for information about issues in current releases of MySQL for PCF.

Planning your Deployment

Network Layout

MySQL for PCF supports deployment to multiple availability zones (AZs) on vSphere only. For other infrastructures, make sure to specify only one AZ.

To achieve best uptime, it is best to deploy a load balancer in front of the SQL Proxy nodes. Please see the note in the [proxy](#) section below. When configuring this load balancer, increase the minimum idle timeout if possible, as many load balancers are tuned for short-lived connections unsuitable for long-running database queries. See the [known issue](#) for more details.


Provided below is a table for those that deploy the MySQL service on a different network than Elastic Runtime. Refer to this table to configure

any firewall rules to allow inbound and outbound traffic required by the MySQL service.

Type	Listening service	TCP Port
Inbound/TCP	Service broker	8081
Inbound/TCP	SQL proxy	3306
Inbound/TCP	Proxy health check	1936
Inbound/TCP	Proxy API	8080
Inbound/TCP	Proxy health check	1936
Outbound/TCP	NATS	4222
Internal/TCP	MySQL server	3306
Internal/TCP	Galera	4567
Internal/TCP	Galera health check	9200

Application Security Groups

You must create an [Application Security Group](#) (ASG) for MySQL for PCF in order for applications to access to the service. See [Creating Application Security Groups for MySQL](#) for instructions.

 **Note:** The service will not be usable until an ASG is in place.

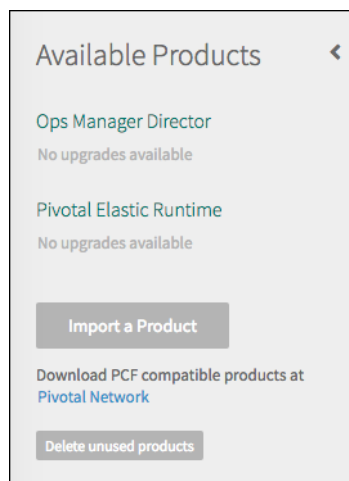
Instance Capacity

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.

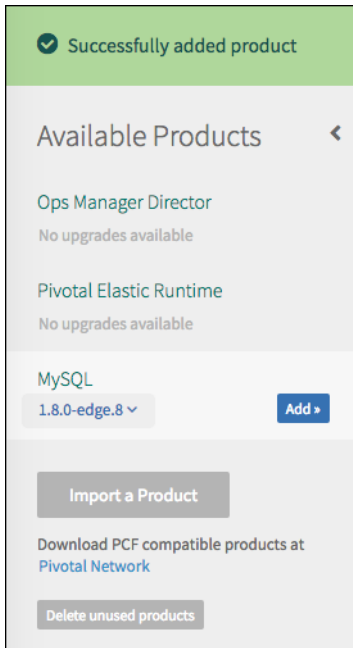
In determining how much persistent disk to make available for databases, operators should also consider that MariaDB servers require sufficient CPU, RAM, and IOPS to promptly respond to client requests for all databases. The MariaDB cluster nodes are configured by default with 100GB of persistent disk. The deployment will fail if this is less than 3GB; we recommend allocating 10GB minimum.

Installation

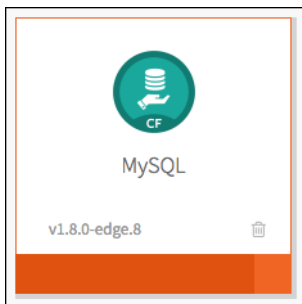
1. Download the product file from [Pivotal Network](#).
2. Navigate to the Ops Manager Installation Dashboard.



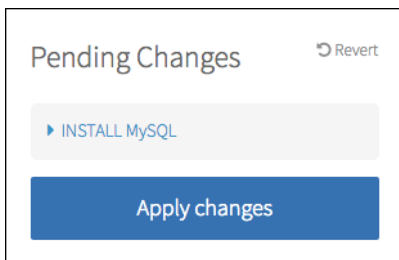
3. Click **Import a Product** to upload the product file to your Ops Manager installation.



4. Click **Add** next to the uploaded product description in the Available Products view to add this product to your staging area.



5. Click the newly added tile to review configurable [Settings](#).



6. Click **Apply Changes** to deploy the service.

Settings

Service Plans

Starting with v1.8.0 of MySQL for PCF, operators can configure multiple service plans. To add or delete plans, follow the instructions below.

Note: If you are upgrading from an earlier version of MySQL for PCF, the upgrade process is not able to continue using the original name of the plan. Upgrading from a version of MySQL for PCF that offered only a single plan causes the plan to be renamed. Regardless of the name of the previous plan (e.g., “100mb-dev”), the plan will now be named `pre-existing-plan`. To retain the same plan name, edit the name before clicking **Apply Changes** to upgrade to MySQL for PCF v1.8.0.

Update Existing Service Instances

You can update service instances using the cf CLI as follows:

```
cf update-service SERVICE_INSTANCE -p NEW_PLAN
```

The following rules apply when updating a service instance plan:


- Updating a service instance to a plan with a larger `max_storage_mb` is always supported.
- Updating a service instance to a plan with a smaller `max_storage_mb` is supported only if the current usage is less than the new value.
- The update command fails if the current usage is greater than the new value.
- Updating a service instance does not disrupt running application connections so long as the application uses fewer than the maximum number of connections allowed by the new plan.


Add a Plan


1. Navigate to the Ops Manager Installation Dashboard and click **MySQL for Pivotal Cloud Foundry**.
2. Click **Service Plans**.
3. Click **Add** to add a new service plan. Click the small triangles to expand or collapse a plan's details.


The screenshot shows the 'MySQL Service Plan Configuration' page. At the top, there's a header 'MySQL Service Plan Configuration'. Below it, a section titled 'Service Plans' includes the text 'Use Service Plans allow you to selectively offer different levels of service to your users' and an 'Add' button. A dropdown menu is open, showing a list of existing plans, with '100mb' selected. Below the dropdown, there's a form to create a new service plan. The form has four required fields: 'Service Plan name', 'Description', 'Storage Quota', and 'Concurrent Connections Quota'. Each field has a text input box. At the bottom of the form, there's a checkbox labeled 'Not available by default'. A 'Save' button is located at the bottom left of the form.

4. Complete the following fields:
 - **Service Plan name:** Plan names may include only lowercase letters, numbers, hyphens, and underscores. Developers use the name of the service plan to create service instances in Apps Manager and the cf CLI.

 **Note:** PCF enforces plan name constraints only at deploy time. The form does not yet allow for real-time validation. If a plan name does not conform to the correct format, the tile fails to deploy after you click **Apply Changes**. The error appears in the **Recent Install Logs** dropdown in the following format: `Error 100: Error filling in template 'settings.yml.erb' for 'cf-mysql-broker-partition-20d9770a220f749796c2/0' (line 40: Plan name 'ONE HUNDRED MEGA BYTES!!!' must only contain lowercase letters, numbers, hyphen(-), or underscore(_)).`

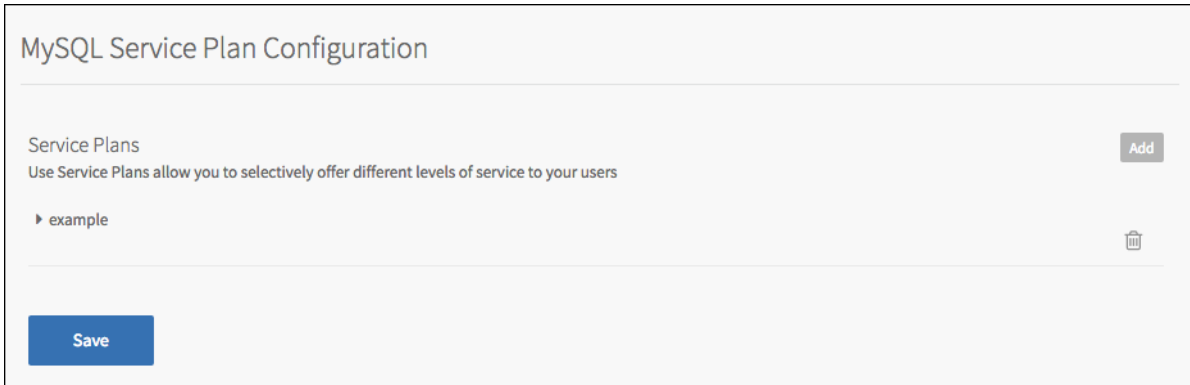
- **Description:** The descriptive text that accompanies the plan name. Use this to provide context beyond what can be expressed by the plan name. For example, “general use, small footprint.”
- **Storage Quota:** The maximum amount, in megabytes, of storage allowed each instance of the Service Plan.
- **Concurrent Connections Quota:** The maximum number of simultaneous database connections allowed to each instance of the Service Plan.
- **Private:** By default, all plans are published to all organizations. Clicking **Not available by default** requires the operator to [publish plans manually](#)  using `cf enable-service-access` .

 **Note:** If you have previously marked a plan as public, and later decide you would like this plan to be private, you need to run `cf disable-service-access` manually for each of your Organizations.


 **Note:** You cannot deploy MySQL for PCF with zero service plans. One plan, at the minimum, is required. If you want to deploy MySQL for PCF without offering any plans, mark the plan as private and do not enable access to any organizations.

Deleting a Plan

1. Navigate to the Ops Manager Installation Dashboard and click the **MySQL for Pivotal Cloud Foundry**.
2. Click **Service Plans**.



3. Click the corresponding trash can icon to delete a plan.

 **Note:** If you accidentally click the trash can, do not click **Save**. Instead, return to the Installation Dashboard and any accidental changes will be discarded. If you do click **Save**, do not click **Apply Changes** on the Installation Dashboard. Instead, click **Revert** to discard any accidental changes.

4. Click **Save**.
5. Click **Apply Changes** from the Installation Dashboard.

If no service instances of the deleted plan exist, the plan will disappear from the Services Marketplace.

If service instances of the deleted plan exist, they will continue to be maintained, but developers cannot create new instances. The service plan will continue to display in the Services Marketplace, marked as “inactive.” Once all service plan instances are deleted, the operator can remove the plan from the Services Marketplace by following these steps:

1. Run `bosh deployments` to find the full name of the MySQL for PCF deployment. For example, `p-mysql-180290d67d5441ebf3c5` .
2. Run `bosh deployment P-MYSQL-DEPLOYMENT-NAME` . For example, `bosh deployment p-mysql-180290d67d5441ebf3c5` .
3. Run `bosh run errand broker-registrar` .

Service Plans before MySQL for PCF v1.8.0


In v1.7 and earlier, the product is only capable of offering one service plan at a time.


A single service plan enforces quotas of 100 megabytes of storage per database and 40 concurrent connections per user by default. Users of Ops 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. Thus, if the storage quota is changed to 1024 megabytes, the new default plan name will be **1024mb-dev**.



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:** After changing a plan's definition, all instances of the plan must be updated. For each plan, either the operator or the user must run `cf update-service SERVICE_INSTANCE -p NEW_PLAN_NAME` on the command line.

 **Note:** Changing a plan's definition does not work properly in versions of MySQL for PCF v1.6.3 and earlier. See the entry in [Known Issues](#) for the recommended workaround.

Options and Features

In the [Advanced Options](#) pane, you can change the configuration of the following features:

- **Disable Reverse DNS lookups**
This feature is enabled by default, and improves performance. Un-checking this option causes the MySQL servers to perform a reverse DNS lookup on each new connection. It is only necessary when restricting access by hostname, which is not required in typical MySQL for PCF installations.
- **Read-Only User Password**
Activates a special user, `roadmin`, a read-only administrator. Supply a special password, to be used only by administrators who require the ability to view all of the data maintained by the MySQL for PCF installation. Leaving the field blank de-activates the read-only user.
- The Replication Canary, see the [monitoring documentation](#).
- The Interruptor, see the [monitoring documentation](#).
- **Quota Enforcer Frequency**
By default, the Quota Enforcer polls for violators and reformers every 30 seconds. This setting, in seconds, changes how long the quota enforcer pauses between checks. If you wish to reduce the small amount of load caused by the Quota Enforcer's polling, you may increase this time period. Be aware, however, that increasing the duration may make it possible for applications to write more data than their pre-determined limit allows.
- **MySQL Start Timeout**
The maximum amount of time necessary for the MySQL process to start, in seconds. When restarting the MySQL server processes, there are conditions under which the process takes longer than expected to appear as running. This can cause parts of the system automation to assume that the process has failed to start properly, and will appear as failing in OpsManager and BOSH output. Depending on the data stored by the database, and the time represented in logs, it may be necessary to increase this beyond the default of 60 seconds.
- **New Cluster Probe Timeout**
There is special logic to detect if a starting node is the first node of a new installation, or if it is part of an existing cluster. Part of this logic is to probe if the other nodes of the cluster have already been deployed. This is set to 30s by default, however due to high latency, this timeout period may be too long. When these probes take too long to respond, it's possible that the **MySQL Start Timeout** may fail the deployment. To account for this, either increase the **MySQL Start Timeout** or lower the **New Cluster Probe Timeout** such that a new node doesn't spend too long performing this test.
- **Replication Debug logging**
By default, the MySQL service will log events related to replication errors. Only turn off this option if error logging is causing undue strain on your logging systems.
- **Server Activity Logging**
The MySQL service includes the [MariaDB Audit plugin](#) . You can disable this plugin, or configure which [events](#)  are recorded. The log can be found at `/var/vcap/store/mysql_audit_logs/mysql_server_audit.log` on each VM. When enabled, the file is rotated every 100 megabytes, and 30 of the most recent files are retained.

 **Note:** Due to the sensitive nature of these logs, they are not transmitted to the syslog server.

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 [Application 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.

Note 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 v1.5.0, existing applications will continue to function, but, 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 v1.5.0 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 apply 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 IP addresses 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 IP addresses are in use.
- In the **Binding Credentials Hostname** field, enter the hostname or IP address that should be given to bound applications for connecting to databases managed by the service. This hostname or IP address 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 the known issue on [proxy behavior](#). Additionally, increase the idle timeout of your load balancer to accommodate long running query times. For more information, see the known issue on [load balancer timeouts](#).

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 v1.5.0 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 rebinding, we recommend configuring a load balancer before deploying v1.5.0.
- 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.

Note: You might also notice a **broker-deregistrar** errand. **Do not run this errand unless instructed to do so by Support.** Broker-deregistrar is a part of the automation used by Ops Manager while deleting a tile. Running this errand under any other circumstance will delete user data.

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 about the 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 Instance Dashboard

Developers can check their current storage usage and service plan quota in the service instance dashboard. You can access this dashboard by either navigating to it from Apps Manager or obtaining its URL from the Cloud Foundry Command-Line Interface (cf CLI):

- **From Apps Manager**

1. Select the space that the service instance runs in.
2. Select the **Services** tab.
3. Under **Services** click the service instance to check.
4. Click **Manage** at top right to open the service instance dashboard.

- **From the cf CLI**


1. Log into the space that the service instance runs in.
2. Run `cf service INSTANCE-NAME`:

```
$ cf service acceptDB
Service instance: acceptDB
Service: p-mysql
Plan: 100mb-dev
Description: MySQL service for application development and testing
Documentation url:
Dashboard: https://p-mysql.sys.acceptance.cf-app.example.com/manage/instances/ddfa6842-b308-4983-a544-50b3d1fb62f0
```

3. Navigate to the URL listed in the output as `Dashboard`. In the example above, the instance dashboard URL is

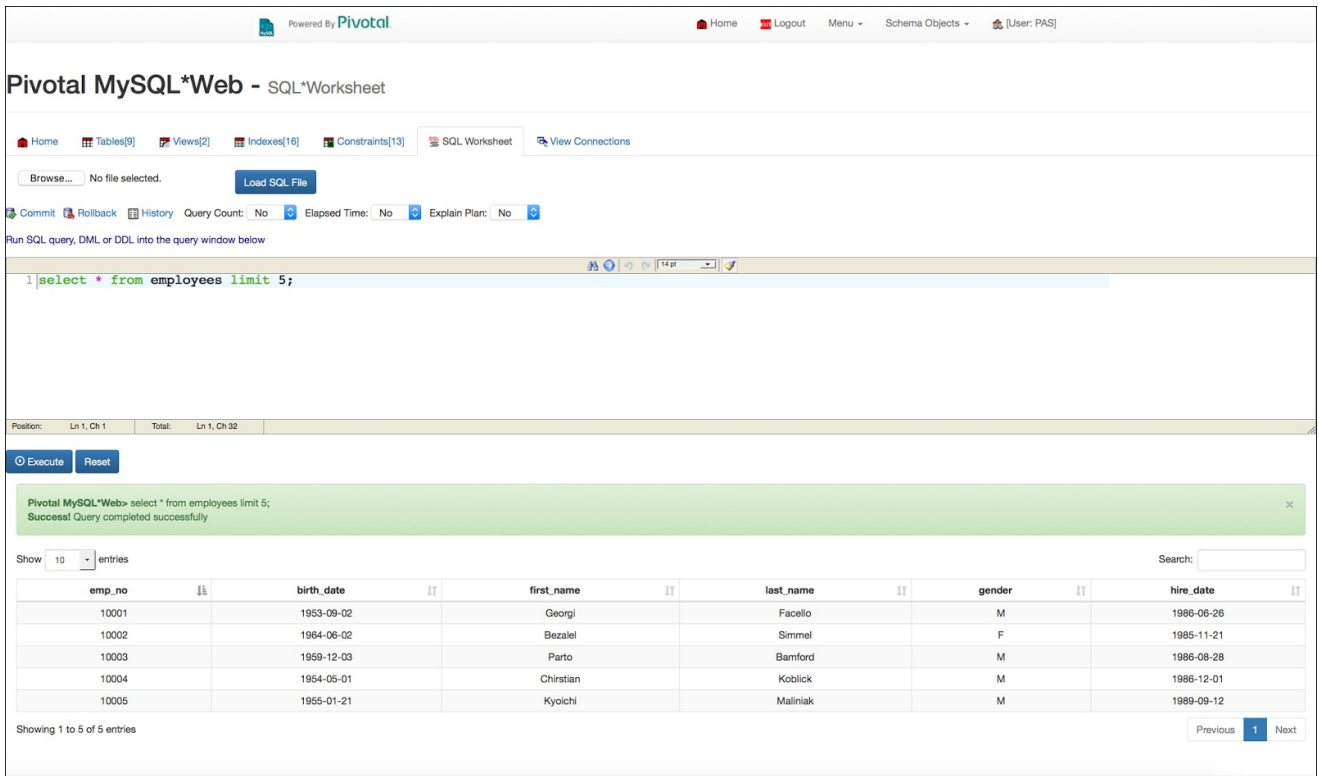
```
https://p-mysql.sys.acceptance.cf-app.example.com/manage/instances/ddfa6842-b308-4983-a544-50b3d1fb62f0
```

The MySQL for PCF service instance dashboard shows how much storage the instance currently uses and the maximum usage allowed by its service plan. It does not show or manage database contents. You can think of it as a gas gauge, while the [Pivotal MySQL Database Management App](#) provides an interface through which you can drive.

 **Note:** The service instance dashboard is distinct from the [proxy dashboard](#) that PCF operators can access to check the proxy instances handling queries for all MySQL for PCF service instances in a PCF deployment.

Pivotal MySQLWeb Database Management App

The Pivotal MySQLWeb app provides a web-based UI for managing MySQL for PCF databases. The free app lets you view and operate on tables, indexes, constraints, and other database structures, and directly execute SQL commands.



You can run the Pivotal MySQLWeb app in two ways:

- Standalone on your own machine
- Deployed to PCF

If you deploy Pivotal MySQLWeb to PCF, you can configure it in the deployment manifest to automatically bind to a specific service instance.

For how to install and run Pivotal MySQLWeb, see the Pivotal MySQLWeb[code repo](#) and [demo video](#)

Command-Line Interface MySQL Plugin

To connect to your MySQL for PCF databases from a command line, use the Cloud Foundry Command-Line Interface (cf CLI) MySQL plugin. The plugin lets you:

- Inspect databases for debugging.
- Manually adjust database schema or contents in development environments.
- Dump and restore databases.

To install the cf CLI MySQL plugin, run the following:

```
$ cf install-plugin -r "CF-Community" mysql-plugin
```

For more information, see the [cf-mysql-plugin](#) repository.

Proxy Dashboard

The service provides a dashboard where administrators can observe health and metrics for each instance in the proxy tier. Metrics include the number of client connections routed to each backend database cluster node.

The dashboard for each proxy instance can be found at `http://proxy-JOB-INDEX-p-mysql.SYSTEM-DOMAIN`.


The job index starts at `0` so if you have two proxy instances deployed and your system-domain is `example.com`, you would access dashboards at `http://proxy-0-p-mysql.example.com` and `http://proxy-1-p-mysql.example.com`.

To access the proxy dashboard, you need basic auth credentials which you can find in the **Credentials** tab of the MySQL for PCF tile in Ops Manager.

For more information about Switchboard, see the [proxy documentation](#).

See Also

- [Cluster Configuration](#)
- [Backing Up MySQL](#)


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

- [Determining Cluster State](#)
- [Cluster Scaling, Node Failure, and Quorum](#)
- [Bootstrapping a Cluster](#)
- [Scaling Down MySQL](#)

Release Notes

v1.8.11

Release Date:

 **This is the last planned release of MySQL for PCF 1.8.** The support period for version 1.8 has expired. To stay up-to-date with the latest software and security updates, please plan to upgrade to MySQL for PCF version 1.9 or greater.

- **Change the Interruptor's default setting to OFF.**

For a [year](#), MySQL for PCF has included [the Interruptor](#). It's a protective mechanism which stops a node from automatically rejoining the cluster if doing so may delete application data. We also upgraded to [MariaDB 10.1](#) and provided the [Replication Canary](#) to further protect application data. There have been zero instances where the Interruptor has been needed to protect application data.

In this release, we are disabling the Interruptor because it is disruptive to normal cluster function, and requires manual Operator action to restore availability. We feel confident that disabling the Interruptor in all but the most critical environments is a safe and convenient choice.

If you wish to continue using the Interruptor, make sure that "Prevent node auto re-join" is checked in the "Advanced Options" configuration pane, then hit **Apply Changes**.

- Upgrades several dependencies including `nokogiri 1.8.0`, `golang 1.8.3`, `xtrabackup 2.4.5`, `boost 1.59.0`, and `python 2.7.13`
- Updated stemcell to 3363.29. This security upgrade resolves the following:
 - [USN-3265-2](#)

For more information, see [pivotal.io/security](#).

v1.8.10

Release Date: June 22, 2017

- New configuration pane for syslog:
Previously, MySQL for PCF used the same configuration settings as Elastic Runtime. However, some users want to send MySQL for PCF logs to destinations other than Elastic Runtime logs. Thus, MySQL for PCF now has separate configuration, similar to RabbitMQ for PCF and Redis for PCF.
Action required: During installation or upgrade of MySQL for PCF, you must configure or disable syslogging in the **Syslog** settings pane.
- Updated stemcell to 3312.29. This security upgrade resolves the following:
 - [USN-3334-1](#)

For more information, see [pivotal.io/security](#).

v1.8.9

Release Date: June 2, 2017

- Updated stemcell to 3312.28. This security upgrade resolves the following:
 - [USN-3291-3](#)

For more information, see [pivotal.io/security](#).

v1.8.8

Release Date: May 19, 2017


- Bug fixes to address issues with MySQL for PCF when the [IPsec add-on](#) is also installed.
 - **Bug fix:** While installing MySQL for PCF with IPsec installed, the product may fail to deploy. This might be due to an issue where the default probe timeout is too long while running under IPsec, and should be reduced. Version 1.8.8 of MySQL for PCF allows you to reduce

the New Cluster Probe Timeout in the MySQL server configuration page. For more information, see [Options and Features](#).

- **Bug fix:** A small change in the way that MySQL nodes shut down, which should better allow nodes to leave the cluster gracefully while IPsec is installed.

v1.8.7



Release Date: April 27, 2017

- Updated stemcell to 3312.24. This security upgrade resolves the following:
 - [USN-3265-2](#) 
- **Bug fix:** Addressed an issue where backups were unable to store backups on AWS S3 regions that require the v4 signature.
- Note: The title of the tile will now appear as “MySQL for PCF,” not simply “MySQL.”

Additional information can be found at <https://pivotal.io/security> 

v1.8.6




Release Date: April 3, 2017

- Updated nokogiri to v1.7.1. This is a security upgrade that resolves the following:
 - [USN-3235-1](#) 
- Updated stemcell to 3263.22. This is a security upgrade that resolves the following:
 - [USN-3249-2](#) 

Additional information can be found at <https://pivotal.io/security> 


v1.8.5

Release Date: March 10, 2017

- **Bug fix:** Changed the value of `wsrep_max_ws_rows` to 0 to prevent MariaDB bug [MDEV-11817](#)  from affecting DDLs.
- **Bug fix:** The server variable, `innodb_large_prefix`, can be disabled via the **Advanced Options** configuration pane. This allows operators to avoid application errors due to [MDEV-12210](#) , a bug in MariaDB 10.1.18. Please see the documentation on [cluster configuration](#) for more information. This option will not be necessary in `p-mysql 1.9` and greater.
- **Bug fix:** We fixed a condition in which a spontaneously rebooted VM may not automatically rejoin the cluster.
- Updated stemcell to 3263.21. This is a security upgrade that resolves the following:
 - [USN-3220-2](#) 

v1.8.4

Release Date: February 24, 2017

- Updated stemcell to 3263.20. This is a security upgrade that resolves the following:
 - [USN-3208-2](#) 
- **Bug fix:** Also addresses a minor issue in which the Operator cannot specify an IP address as the binding credentials host. Previously, this was restricted only to fully-qualified hostnames.

v1.8.3

Release Date: February 9, 2017

- **Bug fix:** Addresses an issue in which rejoin-unsafe and bootstrap errands fail to run properly.

v1.8.2


Release Date: January 26, 2017

- Updated stemcell to 3263.17, which is a routine patch update to address medium and low security vulnerabilities.

Additional information can be found at <https://pivotal.io/security> 

v1.8.1

Release Date: December 16, 2016


- Updated stemcell to 3263.14. This is a security upgrade that resolves the following:
 - [USN-3156-1](#) 
- **Bug fix:** Introduced a configurable startup timeout, which addresses an issue where high-traffic databases may appear as failing upon restart. See the 'advanced configuration' pane to update.

Additional information can be found at <https://pivotal.io/security> 


v1.8.0

Release Date: December 7, 2016

- **Important Change:** This release requires OpsManager v^{1.8}* or higher.
- MariaDB has been updated to v10.1.18 and Galera to v25.3.18.
- **AWS Multi-AZ support:** p-mysql 1.8.0 now balances across availability zones if provisioned with a network that has multiple subnets.
- **Multiple service plans**

 **Note:** On upgrade from a version of MySQL for PCF that offered only a single plan, the default plan will be renamed. Regardless of the name of the previous plan (e.g., `100mb-dev`), the plan will now be named, `pre-existing-plan`. It's not possible to automatically reset the plan to the former name. If you wish to retain the same plan name, it's fine to edit that plan name before clicking **Apply Changes** when upgrading to MySQL for PCF v1.8.0. See [the documentation](#) for more information.

- **New Feature:** `bosh` bootstrap errand
- New `Advanced Options` settings page
 - Skip reverse DNS resolution when accepting connections
This option improves performance, and is only necessary when restricting access by hostname, which is not required in typical MySQL for PCF installations.
 - Includes the option to enable a read-only user.
 - Quota Enforcer check frequency is now configurable.
- **Logging Changes:**
 - **Server Audit** and replication debug logging can now be enabled in the `Advanced Options` settings page.
 - MySQL job logs are kept local on the VM, in addition to sent to syslog if configured.
 - Binary logs are now enabled and rotated automatically by the system.
 - Plus a host of debug log changes have been added to aid in diagnosis efforts.
- MySQL Server configuration changes:
 - XA Transactions are now explicitly disallowed.
 - New service bindings and keys will **not be able to lock tables**.
XA Transactions and table-level locks are not compatible with our HA technology.

 **Note:** Deprecation Notice: This release does not remove the privilege to lock tables from existing bindings, so it won't break any applications that are currently deployed.

• Backups:


- **New feature:** Backing up all nodes

In the Backups configuration pane, there's now an option to take backups from all MySQL nodes. This feature protects your users from data loss in the case that some nodes have different data than the others.

- Automated backups can now additionally target a Ceph back-end storage service or direct to another host via SCP.
- Operator can now override the default number of open files while taking backups.

• New Protections:

We've discovered a rare condition where a MySQL cluster experiences a fault in replication that can result in some data loss. When this occurs, previous releases do not log the root cause of the bug. In order to best address this issue, v1.7.11 contains significant additional telemetry and several defensive features which will account for the failure condition and prevent data loss.


 **Note:** If any of these protections activate, it is critical that you contact Pivotal support immediately. Support will work with you to determine the nature of the cluster's failure, and advise a suggested resolution. Additionally, contacting Support will provide us with evidence that will enable us to identify and address the root cause in the future.

◦ Introducing the Replication Canary

We've included a new long-running monitor, the **Replication Canary**. The Replication Canary continually monitors the MySQL cluster, watching for instances in which cross-cluster replication has failed. It is enabled by default, and requires an e-mail address in the Advanced Options configuration pane.

In the event that replication has failed, the Canary performs two actions:

- E-mail the Operator: Part of the Replication Canary's configuration is an e-mail address, which can be directed to any Operator e-mail address, or an escalation system similar to PagerDuty.
- Deny Access: When replication has failed, the Replication Canary will automatically disable user and applications' ability to access the cluster via the Proxies.

 **Note:** Due to the serious nature of a failure in replication, both behaviors are enabled by default. During configuration, you may elect to set the Replication Canary to notify-only mode, but this is not recommended.

- You **must** set the `Monitoring` job to 1 in the Resource Config pane, or the Replication Canary will not be enabled, regardless of configuration.
- You **must** also confirm that the Elastic Runtime tile is properly configured to send e-mail. These settings are necessary for any standard Cloud Foundry configuration.
 - Ensure that the `Notifications` errand has been enabled.
 - Ensure that `SMTP Config` has been properly configured.


If either of these are not set, configure and **Apply Changes** before deploying v1.7.11.

For more information about the Replication Canary, see the [monitoring documentation](#).

◦ Introducing the Interruptor

The MySQL nodes have new logic that, when enabled, will prevent a node from re-joining a cluster under certain conditions. This is a second level of protection against the possibility of data loss.

For more information about the Interruptor, see the [monitoring documentation](#).

- **Bug fix:** Update `broker-registrar` to avoid runaway CPU condition on broker VMs.
- **Bug fix:** When deployed on Ops Manager 1.7, the `backup-prepare-node` now requires persistent disk instead of a VM with a large amount of CPUs, RAM and ephemeral disk space.
- **Bug fix:** Quota Enforcer should not block other queries from running.
- **Known Issue:** Users with very large databases, depending on performance, will suffer an issue where the system fails to wait sufficiently long to the mysql server to start. This will be resolved in a coming release by introducing a tunable timeout.
- **Security fix:** Switchboard no longer exposes the profiling port by default.
- **Security fix:** The service broker should not use root credentials to access MySQL
- Updated stemcell to 3263.12 to resolve the following:
 - [USN-3151-2](#) 

Additional information can be found at <https://pivotal.io/security> 

Known Issues

Bootstrap and Rejoin-Unsafe Errand Issue

There is an issue in versions 1.8.0 - 1.8.2 in which the `bootstrap` and `rejoin-unsafe` errands fail with an error like this:

```
Error 100: Unable to render jobs for instance group 'rejoin-unsafe'. Errors are:  
- Unable to render templates for job 'rejoin-unsafe'. Errors are:  
- Error filling in template 'config.yml.erb' (line 8: Can't find link 'arbitrator')
```


This is a bug in those releases. It will be addressed in a coming release. In the meanwhile, it may be necessary to perform these steps manually:

- For `bootstrap`, you must use the [manual bootstrap](#) instructions to restore access to the cluster.
- For `rejoin-unsafe`, you must follow these steps:
 1. Log into the node which has tripped the [interruptor](#) and become root.
 2. `monit stop mariadb_ctrl`
 3. `rm -rf /var/vcap/store/mysql`
 4. `/var/vcap/jobs/mysql/bin/pre-start`
 5. `monit start mariadb_ctrl`

Compile fails in environments that do not have access to the Internet

In MySQL for Pivotal Cloud Foundry (PCF) `1.8.0-Edge.5` and `1.8.0-Edge.6`, there is a regression which will cause the compile stage to fail while installing the tile on environments that do not have access to the Internet. We regret the error.

MySQL Backups to AWS S3 limited to Standard region

In MySQL for PCF 1.7, backups are only sent to AWS S3 buckets that have been created in the [US Standard](#)  region, “us-east-1.” This limitation has been resolved in 1.8.0-Edge.2 and later.

Elastic Runtime HTTPS-only feature

Support for the Experimental HTTPS-only feature is broken in MySQL for PCF versions 1.6.X and earlier. The HTTPS-only feature works as designed in MySQL for PCF 1.7.0 and later.

Accidental deletion of a Service Plan

If and only if the Operator does all of these steps in sequence, a plan will become “unrecoverable”:

1. Click the trash-can icon in the Service Plan screen
2. Enter a plan with the exact same name
3. Click the ‘Save’ button on the same screen
4. Return to the Ops Manager top-level, and click ‘Apply Changes’

After clicking ‘Apply Changes’, the deploy will eventually fail with the error:

```
Server error, status code: 502, error code: 270012, message: Service broker catalog is invalid: Plan names must be unique within a service
```

This unfortunate situation is unavoidable; once the Operator has committed via ‘Apply Changes’, the original plan cannot be recovered. For as long as service instances of that plan exist, you may not enter a new plan of the same name. At this point, the only workaround is to create a new plan with the same specifications, but specify a different name. Existing instances will continue to appear under the old plan name, but new instances will need to be created using the new plan name.

If you have committed steps 1 and 2, but not 4, no problem. Do not hit the 'Save' button. Simply return to the Installation Dashboard. Any accidental changes will be discarded.

If you have committed steps 1, 2 and 3, do not click 'Apply Changes.' Instead, return to the Installation Dashboard and click the **Revert** button. Any accidental changes will be discarded.

Changing service plan definition

In MySQL for PCF versions 1.7.0 and earlier, there is only one service plan. Changing the definition of that plan, the number of megabytes, number of connections, or both, will make it so that any new service instances will have those characteristics.

There is a bug in MySQL for PCF versions 1.6.3 and earlier. Changing the plan does not change existing service instances. Existing plans will continue to be governed by the plan constraints effective at the time they were created. This is true regardless of whether or not an operator runs `cf update-service`.

There is a workaround for this bug, which will be resolved in future releases of MySQL for PCF. In order for the change to be effective for existing plans, you must trigger this by interacting directly with the service broker:

```
curl -v -k -X PATCH https://BROKER_CREDS_USERNAME:BROKER_CREDS_PASSWORD@p-  
mysql.SYSTEM.DOMAIN.example.com/v2/service_instances/SERVICE_INSTANCE_ID?plan_id=17d793e6-6da6-4f0e-b58d-364a407166a0
```

- SYSTEM.DOMAIN is defined in Ops Manager, under Elastic Runtime's **Settings** tab, in the `Cloud Controller` entry.
- BROKER_CREDS_USERNAME and BROKER_CREDS_PASSWORD are defined in Ops Manager, under MySQL for PCF's **Credentials** tab, in the `Broker Auth Credentials` entry.
- To get each SERVICE_INSTANCE_ID, run `cf service INSTANCE --guid`. You should see output like this example:

```
$ cf service acceptDB --guid  
4cae3a5e-66b1-4c9a-8536-feaff25237bf
```

Run this for each service instance to be updated.

Furthermore, if you have changed the max number of connections constraint, then it is necessary to update each bound application's setting directly from the MySQL console. Follow these steps:

1. SSH into your Ops Manager Director using these [instructions](#).
2. Run `bosh deployments` to discover the name of your MySQL for PCF deployment.
3. Run `bosh ssh` using your MySQL for PCF's deployment name. Example: `bosh ssh mysql-partition-9d32f5601988152e869b/0`
4. Run `/var/vcap/packages/mariadb/bin/mysql -u root -p`.
 - The root user's password is defined in Ops Manager, under MySQL for PCF's **Credentials** tab.
5. Issue this MySQL command:

```
UPDATE mysql.user SET mysql.user.max_user_connections=NEW_MAX_CONN_VALUE WHERE mysql.user.User NOT LIKE '%root%';
```

 - Make sure to change `NEW_MAX_CONN_VALUE` to whatever new setting you've chosen.
6. `exit;`

Proxies may write to different MySQL masters

All proxy instances use the same method to determine cluster health. However, certain conditions may cause the proxy instances to route to different nodes, for example after brief cluster node failures.

This could be an issue for tables that receive many concurrent writes. Multiple clients writing to the same table could obtain locks on the same row, resulting in a deadlock. One commit will succeed and 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**.

Number of proxy instances cannot be reduced

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.

Backups Metadata

In MySQL for PCF 1.7.0, both `compressed` and `encrypted` show as `N` in the backup metadata file. This is due to the fact that MySQL for PCF implements compression and encryption outside of the tool used to generate the file. This is a known defect, and will be corrected in future releases.

MyISAM tables

The clustering plugin used in this release (Galera) does not support replication of MyISAM Tables. However, the service does not prevent the creation of MyISAM tables. When MyISAM tables are created, the tables will be created on every node (DDL statements are replicated), but data written to a node won't be replicated. If the persistent disk is lost on the node where data is written to (for MyISAM tables only), data will be lost. To change a table from MyISAM to InnoDB, follow this [guide](#).

Max user connections

When updating the `max_user_connections` property for an existing plan, the connections currently open will not be affected. For example, if you have decreased from 20 to 40, users with 40 open connections will keep them open. To force the changes upon users with open connections, an operator can restart the proxy job. This will cause the connections to reconnect and stay within the limit. Otherwise, if any connection above the limit is reset, it won't be able to reconnect, so the number of connections will eventually converge on the new limit.

Long SST transfers

We provide a `database_startup_timeout` in our manifest which specifies how long to wait for the initial SST to complete (default is 150 seconds). If the SST takes longer than this amount of time, the job will report as failing. Versions before `cf-mysql-release v23` have a flaw in our startup script where it does not kill the mysqld process in this case. When monit restarts this process, it sees that mysql is still running and exits without writing a new pidfile. This means the job will continue to report as failing. The only way to fix this is to SSH onto the failing node, kill the mysqld process, and re-run

```
monit start
mariadb_ctrl
```

Long-running queries can be interrupted by load balancer timeout

A connection that is waiting for results will appear to some load balancers as an idle connection. These long-running queries may be interrupted if they exceed the load balancer's idle timeout. The following error is typical of such an interruption:

```
Lost connection to MySQL server during query
```

For example, [AWS's Elastic Load Balancer](#) has a default idle timeout of 60 seconds, so if a query takes longer than this duration then the MySQL connection will be severed and an error will be returned.

To prevent these timeouts, increase the idle timeout duration accordingly.

Frequently Asked Questions

Many replication errors in the logs

I see lots of replication errors in my logs! Is the cluster broken?

Unless the GRA files show a clear execution error (e.g., out of disk space) this is a normal behavior, and it's nothing to worry about. We will be working on more advanced monitoring to detect the failure case, and alert Operators in the future.

Occasionally, you'll see replication errors in the MySQL logs that will look something like this:

```
160318 9:25:16 [Warning] WSREP: RBR event 1 Query apply warning: 1, 16992456 160318 9:25:16 [Warning] WSREP: Ignoring error for TO isolated action: source: abcd1234-abcd-1234-abcd-1234abcd1234 version: 3 local: 0 state: APPLYING flags: 65 conn_id: 246804 trx_id: -1 seqnos (l: 865022, g: 16992456, s: 16992455, d: 16992455, ts: 2530660989030983) 160318 9:25:16 [ERROR] Slave SQL: Error 'Duplicate column name 'number'' on query. Default database: 'cf_0123456_1234_abcd_1234_abcd1234abcd'. Query: 'ALTER TABLE ...'
```

What this is saying is that someone (probably an app) issued an "ALTER TABLE" command that failed to apply to the current schema. More often than not, this is user error.

The node that receives the request processes it as any MySQL server will, if it fails, it just spits that failure back to the app, and the app needs to decide what to do next. That part is normal. HOWEVER, in a Galera cluster, all DDL is replicated, and all replication failures are logged. So in this case, the bad ALTER TABLE command will be run by both slave nodes, and if it fails, those slave nodes will log it as a "replication failure" since they can't tell the difference.

It's really hard to get a valid DDL to work on some nodes, yet fail on others. Usually those cases are limited to out of disk space or working memory. We haven't duplicated that yet.

But I found a blog article that suggests that the schemata can get out of sync?

<https://www.percona.com/blog/2014/07/21/a-schema-change-inconsistency-with-galera-cluster-for-mysql/> 

The key thing about this post is that he had to deliberately switch a node to RSU, which MySQL for Pivotal Cloud Foundry (PCF) never does except during SST. So this is a demonstration of what is possible, but does not explain how a customer may actually experience this in production.

MySQL has blacklisted its own proxy?


What does the error, `blocked because of many connection errors` mean?

There are times when MySQL will blacklist its own proxies:

```
OUT 07:44:02.070 [paasEnv=MYPASS orgName=MYORG spaceName=MYSPACE appName=dc-routing appId=0123456789] [http-nio-8080-exec-5] ERROR o.h.e.jdbc.spi.SqlExceptionHelper - Host '192.0.2.15' is blocked because of many connection errors; unblock with 'mysqladmin flush-hosts'
```

You can solve this by running the following on any of the MySQL job VMS:

```
/var/vcap/jobs/mysql/packages/mariadb/bin/mysqladmin flush-hosts
```

This is an artifact of an automatic polling-protection [feature](#)  built into MySQL and MariaDB. It is a historical feature intended to block Denial of Service attacks. It is usually triggered by a Load Balancer or System Monitoring software performing empty "port checks" against the MySQL proxies. This is why it is important to configure any Load Balancer to perform TCP checks against the proxy health-check port, default **1936**. Repeated port checks against **3306** will cause an outage for all MySQL for PCF users.

- Note: This issue has been disabled as of MySQL for PCF v1.8.0-edge.4.

Cluster Scaling, Node Failure, and Quorum

Documented here are scenarios in which the size of a cluster may change, how the cluster behaves, and how to restore service function when impacted. [Galera Cluster](#) is used to manage the [MariaDB](#) cluster in our release.

Healthy Cluster

Galera documentation refers to nodes in a healthy cluster as being part of [primary component](#). These nodes will respond normally to all queries, reads, writes, and database modifications.

If an individual node is unable to connect to the rest of the cluster (ex: network partition) it becomes non-primary (stops accepting writes and database modifications). In this case, the rest of the cluster should continue to function normally. A non-primary node may eventually regain connectivity and rejoin the primary component.

If more than half of the nodes in a cluster are no longer able to connect to each other, all of the remaining nodes lose quorum and become non-primary. In this case, the cluster must be manually restarted, as documented in the [bootstrapping docs](#).

Graceful removal of a node

- Shutting down a node with `monit` (or decreasing cluster size by one) will cause the node to gracefully leave the cluster.
- Cluster size is reduced by one and maintains healthy state. Cluster will continue to operate, even with a single node, as long as other nodes left gracefully.

Adding new nodes

When new nodes are added to or removed from a MySQL service, a top-level property is updated with the new nodes' IP addresses. As BOSH deploys, it will update the configuration and restart all of the MySQL nodes **and** the proxy nodes (to inform them of the new IP addresses as well). Restarting the nodes will cause all connections to that node to be dropped while the node restarts.

Scaling the cluster

Scaling up from 1 to N nodes

When a new MariaDb node comes online, it replicates data from the existing node in the cluster. Once replication is complete, the node will join the cluster. The proxy will continue to route all incoming connections to the primary node while it remains healthy.

If the proxy detects that this node becomes [unhealthy](#), it will sever existing connections, and route all new connections to a different, healthy node. If there are no healthy MariaDb nodes, the proxy will reject all subsequent connections.

While transitioning from one node to a cluster, there will be an undetermined period of performance degradation while the new node syncs all data from the original node.

Note: If you are planning to scale up MariaDb nodes, it is recommended to do so in different Availability Zones to maximize cluster availability. An Availability Zone is a network-distinct section of a given Region. Further details are available in [Amazon's documentation](#).

Scaling down from N to 1 node


When scaling from multiple nodes to a single MariaDb node, the proxy will determine that the sole remaining node is the primary node (provided it remains healthy). The proxy routes incoming connections to the remaining MariaDb node.

Rejoining the cluster (existing nodes)

Existing nodes restarted with `monit` should automatically join the cluster. If an existing node fails to join the cluster, it may be because its transaction record's (`seqno`) is higher than that of the nodes in the cluster with quorum (aka the primary component).

- If the node has a higher `seqno` it will be apparent in the error log `/var/vcap/sys/log/mysql/mysql.err.log`.
- If the healthy nodes of a cluster have a lower transaction record number than the failing node, it might be desirable to shut down the healthy nodes and bootstrap from the node with the more recent transaction record number. See the [bootstrapping docs](#) for more details.
- Manual recovery may be possible, but is error-prone and involves dumping transactions and applying them to the running cluster (out of scope for this doc).
- Abandoning the data is also an option, if you're ok with losing the unsynchronized transactions. Follow the following steps to abandon the data (as root):
 - Stop the process with `monit stop mariadb_ctrl`.
 - Delete the galera state (`/var/vcap/store/mysql/grastate.dat`) and cache (`/var/vcap/store/mysql/galera.cache`) files from the persistent disk.
 - Restarting the node with `monit start mariadb_ctrl`.

State Snapshot Transfer (SST)

When a new node is added to the cluster or rejoins the cluster, it synchronizes state with the primary component via a process called SST. A single node from the primary component is chosen to act as a state donor. By default Galera uses rsync to perform SST, which blocks for the duration of the transfer. However, MySQL for Pivotal Cloud Foundry (PCF) is configured to use [Xtrabackup](#) , which allows the donor node to continue to accept reads and writes.

Quorum

- In order for the cluster to continue accepting requests, a quorum must be reached by peer-to-peer communication. More than half of the nodes must be responsive to each other to maintain a quorum.
- If more than half of the nodes are unresponsive for a period of time the nodes will stop responding to queries, the cluster will fail, and bootstrapping will be required to re-enable functionality.

Avoid an even number of nodes

- It is generally recommended to avoid an even number of nodes. This is because a partition could cause the entire cluster to lose quorum, as neither remaining component has more than half of the total nodes.
- A 2 node cluster cannot tolerate the failure of single node failure as this would cause loss of quorum. As such, the minimum number of nodes required to tolerate single node failure is 3.

Unresponsive node(s)

- A node can become unresponsive for a number of reasons:
 - network latency
 - mysql process failure
 - firewall rule changes
 - vm failure
- Unresponsive nodes will stop responding to queries and, after timeout, leave the cluster.
- Nodes will be marked as unresponsive (inactive) either:
 - If they fail to respond to one node within 15 seconds
 - OR If they fail to respond to all other nodes within 5 seconds
- Unresponsive nodes that become responsive again will rejoin the cluster, as long as they are on the same IP which is pre-configured in the gcomm address on all the other running nodes, and a quorum was held by the remaining nodes.
- All nodes suspend writes once they notice something is wrong with the cluster (write requests hang). After a timeout period of 5 seconds, requests to non-quorum nodes will fail. Most clients return the error: `WSREP has not yet prepared this node for application use`. Some clients may instead return `unknown error`. Nodes who have reached quorum will continue fulfilling write requests.
- If deployed using a proxy, a continually inactive node will cause the proxy to fail over, selecting a different MySQL node to route new queries to.

Re-bootstrapping the cluster after quorum is lost

- The start script will currently bootstrap node 0 only on initial deploy. If bootstrapping is necessary at a later date, it must be done manually. For more information about manually bootstrapping a cluster, see [Bootstrapping Galera](#).
- If the single node is bootstrapped, it will create a new one-node cluster that other nodes can join.

Simulating node failure

- To simulate a temporary single node failure, use `kill -9` on the pid of the MySQL process. This will only temporarily disable the node because the process is being monitored by monit, which will restart the process if it is not running.
- To more permanently disable the process, execute `monit unmonitor mariadb_ctrl` before `kill -9`.
- To simulate multi-node failure without killing a node process, communication can be severed by changing the iptables config to disallow communication:

```
iptables -F && # optional - flush existing rules \  
iptables -A INPUT -p tcp --destination-port 4567 -j DROP && \  
iptables -A INPUT -p tcp --destination-port 4568 -j DROP && \  
iptables -A INPUT -p tcp --destination-port 4444 -j DROP && \  
iptables -A INPUT -p tcp --destination-port 3306 && \  
iptables -A OUTPUT -p tcp --destination-port 4567 -j DROP && \  
iptables -A OUTPUT -p tcp --destination-port 4568 -j DROP && \  
iptables -A OUTPUT -p tcp --destination-port 4444 -j DROP && \  
iptables -A OUTPUT -p tcp --destination-port 3306
```

To recover from this, drop the partition by flushing all rules: `iptables -F`

Cluster Configuration

This page documents the various configuration decisions that have been made in relation to MariaDB and Galera in cf-mysql-release.

SST method

Galera supports multiple methods for [State Snapshot Transfer](#) [\[3\]](#). The `rsync` method is usually fastest. The `xtrabackup` method has the advantage of keeping the donor node writeable during SST. We have chosen to use `xtrabackup`.

InnoDB Log Files

Our cluster defaults to 1GB for log file size to support larger blob.

Max User Connections

To ensure all users get fair access to system resources, we default each user's number of connections to 40. Operators can override this setting, per plan, in the Service Plans configuration pane.

Skip External Locking

Since each Virtual Machine only has one mysqld process running, we do not need external locking.

Max Allowed Packet

We allow blobs up to 256MB. This size is unlikely to limit a user's query, but is also manageable for our InnoDB log file size.

InnoDB File Per Table

InnoDB allows using either a single file to represent all data, or a separate file for each table. We chose to use a separate file for each table as this provides more flexibility and optimization. For a full list of pros and cons, see MySQL's documentation for [InnoDB File-Per-Table Mode](#) [\[4\]](#).

InnoDB File Format

To take advantage of all the extra features available with the `innodb_file_per_table = ON` option, we use the `Barracuda` file format.

InnoDB Large Prefix

The `innodb_large_prefix` feature is enabled by default. When disabled, large index prefixes are silently truncated. When enabled, larger index key prefixes may be created by additionally specifying DYNAMIC or COMPRESSED row formats. Row format is not defined by default. Thus, when `innodb_large_prefix` is enabled, applications must specify row format for each table which will use large index prefixes.

Slow Query Log

MySQL for PCF automatically enables the [slow query log](#) [\[5\]](#) and sets it to record any query that takes longer than 10 seconds. By default, those logs appear in the file `/var/vcap/sys/log/mysql/mysql_slow_query.log` on each node. For a consolidated view, use a syslog server.

Temporary Tables

MySQL is configured to convert temporary in-memory tables to temporary on-disk tables when a query EITHER generates more than 16 million rows of output or uses more than 32MB of data space. Users can see if a query is using a temporary table by using the EXPLAIN command and looking for “Using temporary,” in the output. If the server processes very large queries that use /tmp space simultaneously, it is possible for queries to receive no space left errors.

Reverse Name resolution

Since our connection authentication is implemented using user credentials, the default implementation does not restrict to host names from which a user may connect. To save time on each new connection, we’ve turned off reverse name resolution by default.

Large Data File Ingestion

Now that a major bug has been fixed in MariaDB, we enable `wsrep_load_data_splitting` which splits large data imports into separate transactions to enable loading data into a MariaDB cluster.

Proxy for MySQL for Pivotal Cloud Foundry

In MySQL for Pivotal Cloud Foundry (PCF), [Switchboard](#) is used to proxy TCP connections to healthy MariaDB nodes.

A proxy is used to gracefully handle failure of MariaDB nodes. Use of a proxy permits very fast, unambiguous failover to other nodes within the cluster in the event of a node failure.

When a node becomes unhealthy, the proxy re-routes all subsequent connections to a healthy node. All existing connections to the unhealthy node are closed.

Proxy Dashboard

The service provides a dashboard where administrators can observe health and metrics for each instance in the proxy tier. Metrics include the number of client connections routed to each backend database cluster node.

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

- <http://proxy-0-p-mysql.example.com>
- <http://proxy-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.

Consistent Routing

At any given time, Switchboard will only route to one active node. That node will continue to be the only active node until it becomes unhealthy.

If multiple Switchboard proxies are used in parallel (ex: behind a load-balancer) there is no guarantee that the proxies will choose the same active node. This can result in deadlocks, wherein attempts to update the same row by multiple clients will result one commit succeeding and the other fails. This is a known issue, with exploration of mitigation options on the roadmap for this product. To avoid this problem, use a single proxy instance or an external failover system to direct traffic to one proxy instance at a time.

Node Health

Healthy

The proxy queries an HTTP healthcheck process, co-located on the database node, when determining where to route traffic.

If the healthcheck process returns HTTP status code of 200, the node is added to the pool of healthy nodes.

A resurrected node will not immediately receive connections. The proxy will continue to route all connections, new or existing, to the currently active node. In the case of failover, all healthy nodes will be considered as candidates for new connections.

Unhealthy

If the healthcheck returns HTTP status code 503, the node is considered unhealthy.

This happens when a node becomes non-primary, as specified by the [cluster-behavior docs](#).

The proxy will sever all existing connections to newly unhealthy nodes. Clients are expected to handle reconnecting on connection failure. The proxy will route new connections to a healthy node, assuming such a node exists.

Unresponsive

If node health cannot be determined due to an unreachable or unresponsive healthcheck endpoint, the proxy will consider the node unhealthy.

This may happen if there is a network partition or if the VM containing the healthcheck and MariaDB node died.

Proxy count

If the operator sets the total number of proxies to 0 hosts in OpsManager or BOSH deployment manifest, then applications will connect directly to one healthy MariaDB node making that node a single point of failure for the cluster.

The recommended number of proxies are 2; this provides redundancy should one of the proxies fail.

Removing the proxy as a SPOF

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

Bound applications are provided with a hostname or IP address to reach a database managed by the service. 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 load balancer

Configure the load balancer to route traffic for TCP port 3306 to the IPs of all proxy instances on TCP port 3306. Next, configure the load balancer's healthcheck to use the proxy health port. This is TCP port 1936 by default to maintain backwards compatibility with previous releases. This port is not configurable.

Configuring MySQL for PCF to give applications the address of the load balancer

To ensure that bound applications will use the load balancer to reach bound databases, navigate to the MySQL for PCF tile in Operations Manager, then the Resource Config configuration screen within it. **On AWS only**, enter your load balancer's hostname in the "ELB Names" column for the Proxy row.

AWS Route 53

To set up a Round Robin DNS across multiple proxy IPs using AWS Route 53, follow the following instructions:

1. Log in to AWS.
2. Click Route 53.
3. Click Hosted Zones.
4. Select the hosted zone that contains the domain name to apply round robin routing to.
5. Click 'Go to Record Sets'.
6. Select the record set containing the desired domain name.
7. In the value input, enter the IP addresses of each proxy VM, separated by a newline.

Finally, update the manifest property `properties.mysql_node.host` for the cf-mysql-broker job, as described above.

API

The proxy hosts a JSON API at `proxy-<bosh job index>.p-mysql.<system domain>/v0/`.

The API provides the following route:

Request:

- Method: GET
- Path: `/v0/backends`
- Params: ~
- Headers: Basic Auth

Response:


```
[
  {
    "name": "mysql-0",
    "ip": "1.2.3.4",
    "healthy": true,
    "active": true,
    "currentSessionCount": 2
  },
  {
    "name": "mysql-1",
    "ip": "5.6.7.8",
    "healthy": false,
    "active": false,
    "currentSessionCount": 0
  },
  {
    "name": "mysql-2",
    "ip": "9.9.9.9",
    "healthy": true,
    "active": false,
    "currentSessionCount": 0
  }
]
```

For more information about SwitchBoard, read the [proxy documentation](#).

Creating Application Security Groups for MySQL

This topic describes how to create [Application Security Groups](#) (ASGs) for MySQL for Pivotal Cloud Foundry (PCF).

To allow smoke tests to run when you install the MySQL for PCF service and allow apps to access MySQL for PCF after it is installed, you must create an appropriate ASG and bind it to the service.

 **Note:** Without an ASG, the service is not installable or usable.

In addition, application containers that access instances of this service require an outbound network connection to the load balancer configured for the MySQL for PCF service.

To create ASGs for the MySQL for PCF service, perform the following steps:

1. Create a JSON file with the following contents called `p-mysql-security-group.json`:

```
[
  {
    "ports": "3306",
    "protocol": "tcp",
    "destination": "REPLACE WITH THE P-MYSQL LOAD BALANCER IP, RANGE OR CIDR"
  }
]
```

In the `destination` field, add the IP address, range, or CIDR of the load balancer that you configured for the MySQL for PCF service.

2. Log in to your PCF deployment as an administrator, and create an ASG named `p-mysql-service`.

```
# after logging in as an administrator
$ cf create-security-group p-mysql-service p-mysql-security-group.json
```

3. Bind the new ASG to the `default-running` ASG set to allow all applications to access the service.

```
$ cf bind-running-security-group p-mysql-service
```

If the service should only be made available to specific spaces, bind the ASG directly to those spaces.

```
$ cf bind-security-group p-mysql-service ORGANIZATION_NAME SPACE_NAME
```

Monitoring the MySQL Service

This document describes how to use the Replication Canary and Interruptor to monitor your MySQL cluster.

Replication Canary


MySQL for Pivotal Cloud Foundry (PCF) is a clustered solution that uses replication to provide benefits such as quick failover and rolling upgrades. This is more complex than a single node system with no replication. MySQL for PCF includes a Replication Canary to help with the increased complexity. The Replication Canary is a long-running monitor that validates that replication is working within the MySQL cluster.

How it Works

The Replication Canary writes to a private dataset in the cluster, and attempts to read that data from each node. It pauses between writing and reading to ensure that the writesets have been committed across each node of the cluster. The private dataset does not use a significant amount of disk capacity.

When replication fails to work properly, the Canary detects that it cannot read the data from all nodes, and immediately takes two actions:

- E-mails a pre-configured address with a message that replication has failed. See the [sample](#) below.
- Disables client [access to the cluster](#).

 **Note:** Malfunctioning replication exposes the cluster to the possibility of data loss. Because of this, both behaviors are enabled by default. It is critical that you contact Pivotal support immediately in the case of replication failure. Support will work with you to determine the nature of the cluster failure and provide guidance regarding a solution.

Sample Notification E-mail

If the Canary detects a replication failure, it immediately sends an e-mail through the Elastic Runtime notification service. See the following example:

```
Subject: CF Notification: p-mysql Replication Canary, alert 417

This message was sent directly to your email address.

{alert-code 417}
This is an e-mail to notify you that the MySQL service's replication canary has detected an unsafe cluster condition in which replication is not performing as expected across all nodes.
```

Cluster Access

Each time the Canary detects cluster replication failure, it instructs all proxies to disable connections to the database cluster. If the replication issue resolves, the Canary detects this and automatically restores client access to the cluster.

If you must restore access to the cluster regardless of the Replication Canary, contact Support.

Determine Proxy State


You can determine if the Canary disabled cluster access by using the Proxy API. See the following example:

```
ubuntu@ip-10-0-0-38:~$ curl -ku admin:PASSWORD_FROM_OPSMGR -X GET https://proxy-0-p-mysql.SYSTEM-DOMAIN/v0/cluster ; echo
{"currentBackendIndex":0,"trafficEnabled":false,"message":"Disabling cluster traffic","lastUpdated":"2016-07-27T05:16:29.197754077Z"}
```

Enable the Replication Canary

To enable the Replication Canary, follow the instructions below to configure both the Elastic Runtime tile and the MySQL for PCF tile.

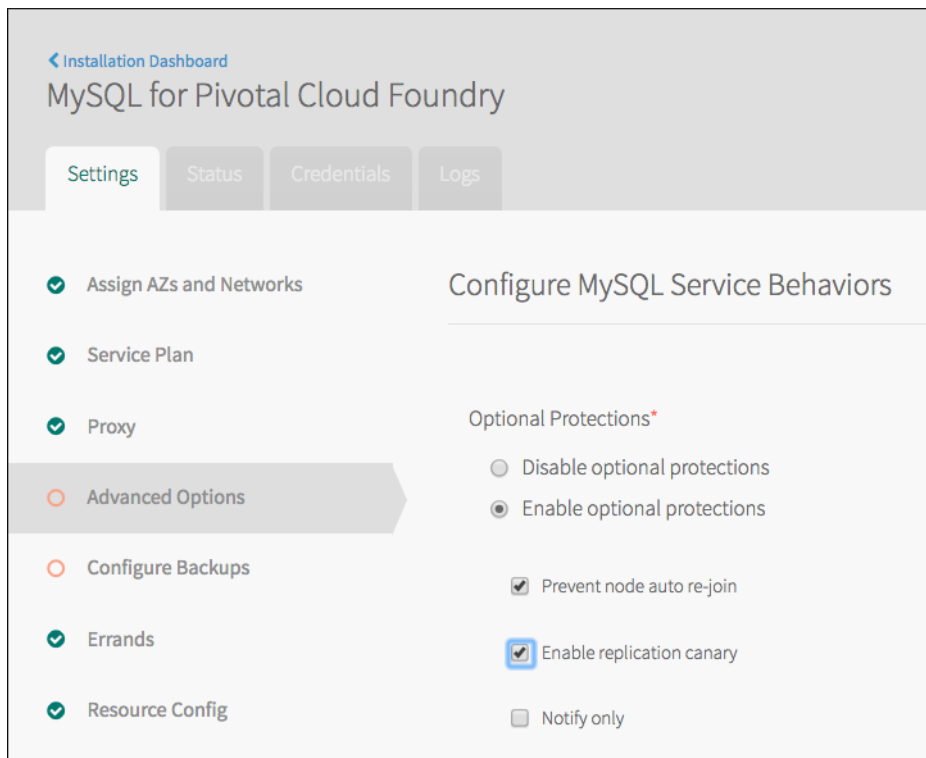
Configure the Elastic Runtime Tile

 **Note:** In a typical PCF deployment, these settings are already configured.

1. In the **SMTP Config** section, enter a **From Email** that the Replication Canary can use to send notifications, along with the SMTP server configuration.
2. In the **Errands** section, select the **Notifications** errand.

Configure the MySQL for PCF Tile

1. In the **Advanced Options** section, select **Enable replication canary**.



Installation Dashboard

MySQL for Pivotal Cloud Foundry

Settings | Status | Credentials | Logs

Assign AZs and Networks ✓

Service Plan ✓

Proxy ✓

Advanced Options

Configure Backups

Errands ✓

Resource Config ✓

Configure MySQL Service Behaviors

Optional Protections*

☐ Disable optional protections

☒ Enable optional protections

☒ Prevent node auto re-join

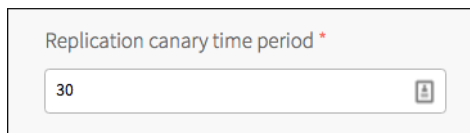
☒ Enable replication canary

☐ Notify only

2. If you want to the Replication Canary to send e-mail but not disable access at the proxy, select **Notify only**.

 **Note:** Pivotal recommends leaving this checkbox unselected due to the possibility of data loss from replication failure.

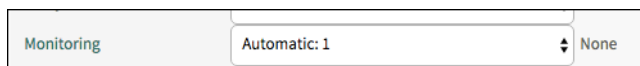
3. You can override the **Replication canary time period**. The **Replication canary time period** sets how frequently the canary checks for replication failure, in seconds. This adds a small amount of load to the databases, but the canary reacts more quickly to replication failure. The default is 30 seconds.



Replication canary time period *

30

4. You can override the **Replication canary read delay**. The **Replication canary read delay** sets how long the canary waits to verify data is replicating across each MySQL node, in seconds. Clusters under heavy load experience some small replication lag as writesets are committed across the nodes. The Default is 20 seconds.
5. Enter an **E-mail address** to receive monitoring notifications. Use a closely monitored e-mail address account. The purpose of the Canary is to escalate replication failure as quickly as possible.
6. In the **Resource Config** section, ensure the **Monitoring** job has one instance.



Monitoring

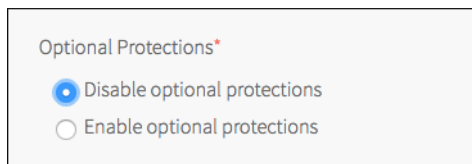
Automatic: 1

None

Disable the Replication Canary

If you do not need the Replication Canary, for instance if you use a single MySQL node, follow this procedure to disable both the job and the resource configuration.

1. In the **Advanced Options** section of the MySQL for PCF tile, select **Disable Replication Canary**.

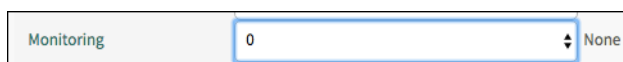


Optional Protections*

☒ Disable optional protections

☐ Enable optional protections

2. In the **Resource Config** pane, set the **Monitoring** job to zero instances.



Monitoring 0 None

Interruptor

There are rare cases in which a MySQL node silently falls out of sync with the other nodes of the cluster. The [Replication Canary](#) closely monitors the cluster for this condition. However, if the Replication Canary does not detect the failure, the Interruptor provides a solution for preventing data loss.

How it Works

If the node receiving traffic from the proxy falls out of sync with the cluster, it generates a dataset that the other nodes do not have. If the same node later receives a transaction that is not compatible with the datasets of the other nodes, it discards its local dataset and adopts the datasets of the other nodes. This is generally desired behavior, unless data replication is not functioning across the cluster. The node could destroy valid data by discarding its local dataset. When enabled, the Interruptor prevents the node from destroying its local dataset if there is a risk of losing valid data.

Note: If you receive a notification that the Interruptor has activated, it is critical that you contact Pivotal support immediately. Support will work with you to determine the nature of the failure, and provide guidance regarding a solution.

An out-of-sync node employs one of two [two modes](#) to catch up with the cluster:

- **Incremental State Transfer (IST):** If a node has been out of the cluster for a relatively short period of time, such as a reboot, the node invokes IST. This is not a dangerous operation, and the Interruptor does not interfere.
- **State Snapshot Transfer (SST):** If a node has been unavailable for an extended amount of time, such as a hardware failure that requires physical repair, the node may invoke SST. In cases of failed replication, SST can cause data loss. When enabled, the Interruptor prevents this method of recovery.

Sample Notification E-mail

The Interruptor sends an email through the Elastic Runtime notification service when it prevents a node from rejoining a cluster. See the following example:

Subject: CF Notification: p-mysql alert 100

This message was sent directly to your email address.

{alert-code 100}

Hello, just wanted to let you know that the MySQL node/cluster has gone down and has been disallowed from re-joining by the interruptor.

Interruptor Logs

You can confirm that the Interruptor has activated by examining `/var/vcap/sys/log/mysql/mysql.err.log` on the failing node. The log contains the following message:

```
WSREP_SST: [ERROR] ##### (20160610 04:33:21.338)
WSREP_SST: [ERROR] SST disabled due to danger of data loss. Verify data and run the rejoin-unsafe errand (20160610 04:33:21.340)
WSREP_SST: [ERROR] ##### (20160610 04:33:21.341)
```

Force a Node to Rejoin the Cluster

In general, if the Interruptor has activated but the Replication Canary has not triggered, it is safe for the node to rejoin the cluster. You can check the health of the remaining nodes in the cluster by following the same [instructions](#) used before scaling from clustered to single-node mode.

Note: This topic requires you to run commands from the [Ops Manager Director](#) using the BOSH CLI. Refer to the [Advanced Troubleshooting with the BOSH CLI](#) topic for more information.

1. Follow these instructions to [choose the p-mysql manifest](#) with the BOSH CLI.
2. Run `bosh run errand rejoin-unsafe` to force a node to rejoin the cluster:

```
$ bosh run errand rejoin-unsafe
[...]
[stdout]
Started rejoin-unsafe errand ...
Successfully repaired cluster
rejoin-unsafe errand completed

[stderr]
None

Errand `rejoin-unsafe` completed successfully (exit code 0)
```

Manual Rejoin

- If the `rejoin-unsafe` errand is not able to cause a node to join the cluster, follow these steps to bypass the Interruptor manually.
 1. Log into each node which has tripped the [interruptor](#) and become root.
 2. `monit stop mariadb_ctrl`
 3. `rm -rf /var/vcap/store/mysql`
 4. `/var/vcap/jobs/mysql/bin/pre-start`
 5. `monit start mariadb_ctrl`

Disable the Interruptor

The Interruptor is enabled by default. To disable the Interruptor:

In the **Advanced Options** section, under **Enable optional protections**, un-check **Prevent node auto re-join**.

☐ Prevent node auto re-join

Determining Cluster State

Connect to each MySQL node using a MySQL client and check its status.

```
$ mysql -h NODE_IP -u root -pPASSWORD -e 'SHOW STATUS LIKE "wsrep_cluster_status";'
+-----+
| Variable_name | Value |
+-----+
| wsrep_cluster_status | Primary |
+-----+
```

If all nodes are in the `Primary` component, you have a healthy cluster. If some nodes are in a `Non-primary` component, those nodes are not able to join the cluster.

See how many nodes are in the cluster.

```
$ mysql -h NODE_IP -u root -pPASSWORD -e 'SHOW STATUS LIKE "wsrep_cluster_size";'
+-----+
| Variable_name | Value |
+-----+
| wsrep_cluster_size | 3 |
+-----+
```

If the value of `wsrep_cluster_size` is equal to the expected number of nodes, then all nodes have joined the cluster. Otherwise, check network connectivity between nodes and use `monit status` to identify any issues preventing nodes from starting.

For more information, see the official Galera documentation for [Checking Cluster Integrity](#).

Bootstrapping a Galera Cluster

Bootstrapping is the process of (re)starting a Galera cluster.

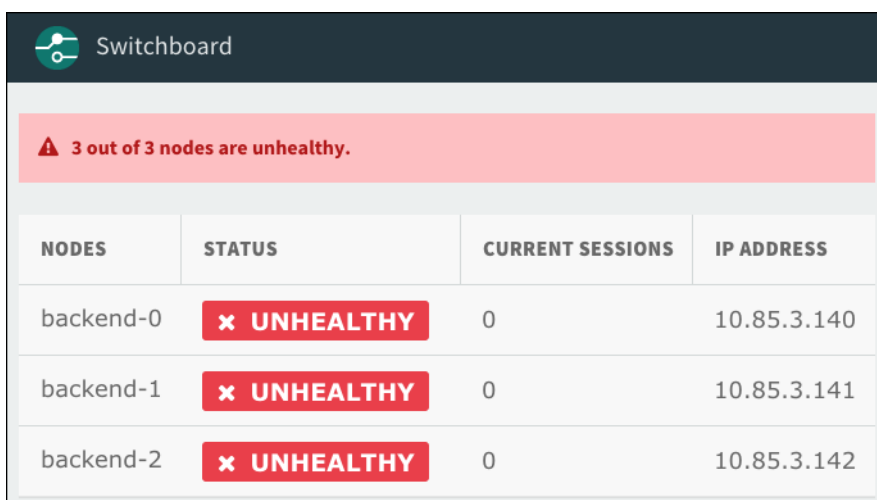
When to Bootstrap

Bootstrapping is only required when the cluster has lost quorum.

Quorum is lost when less than half of the nodes can communicate with each other (for longer than the configured grace period). In Galera terminology, if a node can communicate with the rest of the cluster, its database is in a good state, and it reports itself as `synced`.

If quorum has *not* been lost, individual unhealthy nodes should automatically rejoin the cluster once repaired (error resolved, node restarted, or connectivity restored).

- Symptoms of Lost Quorum
 - All nodes appear “Unhealthy” on the proxy dashboard:



The image shows a screenshot of the Switchboard proxy dashboard. At the top, there is a red alert banner that says "3 out of 3 nodes are unhealthy." Below this, there is a table with four columns: NODES, STATUS, CURRENT SESSIONS, and IP ADDRESS. The table lists three nodes: backend-0, backend-1, and backend-2. All three nodes have a status of "UNHEALTHY" (indicated by a red 'x' icon), 0 current sessions, and IP addresses 10.85.3.140, 10.85.3.141, and 10.85.3.142 respectively.

NODES	STATUS	CURRENT SESSIONS	IP ADDRESS
backend-0	✖ UNHEALTHY	0	10.85.3.140
backend-1	✖ UNHEALTHY	0	10.85.3.141
backend-2	✖ UNHEALTHY	0	10.85.3.142

- 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 responsive nodes respond with `ERROR 1047` when using most statement types:

```
mysql> select * from mysql.user;
ERROR 1047 (08S01) at line 1: WSREP has not yet prepared node for application use
```

See [Cluster Behavior](#) for more details about determining cluster state.

Bootstrapping

Bootstrapping requires you to run commands from the [Ops Manager Director](#). Follow the instructions to use the [BOSH CLI](#) for command-line access.

Note: The examples in these instructions reflect a three-node MySQL for Pivotal Cloud Foundry (PCF) deployment. The process to bootstrap a two-node plus an arbitrator is identical, but the output will not match the examples.

Assisted Bootstrap

1. MySQL for PCF versions 1.8.0 and later include a [BOSH errand](#) to automate the process of bootstrapping. It is still necessary to manually initiate the bootstrap process, but using this errand reduces the number of manual steps necessary to complete the process. In most cases, running the errand is sufficient, however there are some conditions which require additional steps.

How it works

The bootstrap errand simply automates the steps in the manual bootstrapping process documented below. It finds the node with the highest transaction sequence number, and asks it to start up by itself (i.e. in bootstrap mode), then asks the remaining nodes to join the cluster.

Scenario 1: Virtual Machines running, Cluster Disrupted

- a. If the nodes are up and running, but the cluster has been disrupted, the jobs will appear as `failing.` The output of `bosh instances` will look like this:

```
$ bosh instances
[...]
+-----+-----+-----+-----+
| Instance | State | Resource Pool | IPs |
+-----+-----+-----+-----+
| cf-mysql-broker-partition-a813339fde9330e9b905/0 | running | cf-mysql-broker-partition-a813339fde9330e9b905 | 192.0.2.61 |
| cf-mysql-broker-partition-a813339fde9330e9b905/1 | running | cf-mysql-broker-partition-a813339fde9330e9b905 | 192.0.2.62 |
| mysql-partition-a813339fde9330e9b905/0 | failing | mysql-partition-a813339fde9330e9b905 | 192.0.2.55 |
| mysql-partition-a813339fde9330e9b905/1 | failing | mysql-partition-a813339fde9330e9b905 | 192.0.2.56 |
| mysql-partition-a813339fde9330e9b905/2 | failing | mysql-partition-a813339fde9330e9b905 | 192.0.2.57 |
| proxy-partition-a813339fde9330e9b905/0 | running | proxy-partition-a813339fde9330e9b905 | 192.0.2.59 |
| proxy-partition-a813339fde9330e9b905/1 | running | proxy-partition-a813339fde9330e9b905 | 192.0.2.60 |
+-----+-----+-----+-----+
```

In this situation, it is OK to immediately try the bootstrap errand:

- i. Log into the BOSH director
- ii. Select the correct deployment
- iii. `bosh run errand bootstrap`

You will see many lines of output, eventually followed by:

```
Bootstrap errand completed

[stderr]
+ echo 'Started bootstrap errand ...'
+ JOB_DIR=/var/vcap/jobs/bootstrap
+ CONFIG_PATH=/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)
```

There are times when this won't work immediately. Unfortunately, sometimes it is best to wait and try again a few minutes later.

Scenario 2: Virtual Machines Terminated or Lost

- a. In more severe circumstances, such as power failure, it's possible that all of your VMs have been lost. They'll need to be recreated before you can begin to recover the cluster. In this case, you'll see the nodes appear as `unknown/unknown` in the BOSH output.

```
$ bosh instances
```

Instance	State	Resource Pool	IPs
unknown/unknown	unresponsive agent		
unknown/unknown	unresponsive agent		
unknown/unknown	unresponsive agent		
cf-mysql-broker-partition-e97dae91e44681e0b543/0	running	cf-mysql-broker-partition-e97dae91e44681e0b543	192.0.2.65
cf-mysql-broker-partition-e97dae91e44681e0b543/1	running	cf-mysql-broker-partition-e97dae91e44681e0b543	192.0.2.66
proxy-partition-e97dae91e44681e0b543/0	running	proxy-partition-e97dae91e44681e0b543	192.0.2.63
proxy-partition-e97dae91e44681e0b543/1	running	proxy-partition-e97dae91e44681e0b543	192.0.2.64

VM Recovery

VM Recovery is best left to OpsManager by configuring the VM [Resurrector](#) ☒. If enabled, the system will notice that the VMs are gone, and automatically attempt to recreate them. You will be able to see evidence of that by seeing the scan-and-fix job in the output of `bosh tasks recent --no-filter`.

```
$ bosh tasks recent --no-filter
```

#	State	Timestamp	User	Description	Result
123	queued	2016-01-08 00:18:07 UTC	director	scan and fix	

If you have not configured the Resurrector to run automatically, you can also run the BOSH Cloud Check interactive command `bosh cck` to delete any placeholder VMs. If given the option, select **Delete VM reference**.

```

$ bosh cck

Acting as user 'director' on deployment 'cf-e82cbf44613594d8a155' on 'p-bosh-30c19bdd43c55c627d70'
Performing cloud check...

Director task 34
Started scanning 22 vms
Started scanning 22 vms > Checking VM states. Done (00:00:10)
Started scanning 22 vms > 19 OK, 0 unresponsive, 3 missing, 0 unbound, 0 out of sync. Done (00:00:00)
  Done scanning 22 vms (00:00:10)

Started scanning 10 persistent disks
Started scanning 10 persistent disks > Looking for inactive disks. Done (00:00:02)
Started scanning 10 persistent disks > 10 OK, 0 missing, 0 inactive, 0 mount-info mismatch. Done (00:00:00)
  Done scanning 10 persistent disks (00:00:02)

Task 34 done

Started 2015-11-26 01:42:42 UTC
Finished 2015-11-26 01:42:54 UTC
Duration 00:00:12

Scan is complete, checking if any problems found.

Found 3 problems

Problem 1 of 3: VM with cloud ID 'i-afe2801f' missing.
  1. Skip for now
  2. Recreate VM
  3. Delete VM reference
Please choose a resolution [1 - 3]: 3

Problem 2 of 3: VM with cloud ID 'i-36741a86' missing.
  1. Skip for now
  2. Recreate VM
  3. Delete VM reference
Please choose a resolution [1 - 3]: 3

Problem 3 of 3: VM with cloud ID 'i-ce751b7e' missing.
  1. Skip for now
  2. Recreate VM
  3. Delete VM reference
Please choose a resolution [1 - 3]: 3

Below is the list of resolutions you've provided
Please make sure everything is fine and confirm your changes

  1. VM with cloud ID 'i-afe2801f' missing.
    Delete VM reference

  2. VM with cloud ID 'i-36741a86' missing.
    Delete VM reference

  3. VM with cloud ID 'i-ce751b7e' missing.
    Delete VM reference

Apply resolutions? (type 'yes' to continue): yes
Applying resolutions...

Director task 35
Started applying problem resolutions
Started applying problem resolutions > missing_vm 11: Delete VM reference. Done (00:00:00)
Started applying problem resolutions > missing_vm 27: Delete VM reference. Done (00:00:00)
Started applying problem resolutions > missing_vm 26: Delete VM reference. Done (00:00:00)
  Done applying problem resolutions (00:00:00)

Task 35 done

Started 2015-11-26 01:47:08 UTC
Finished 2015-11-26 01:47:08 UTC
Duration 00:00:00
Cloudcheck is finished

```

By watching `bosh instances` you'll see the VMs transition from `unresponsive agent` to `starting`. Ultimately, two will appear as `failing`, this is OK.

```
$ bosh instances
[...]
+-----+-----+-----+-----+
| mysql-partition-e97dae91e44681e0b543/0 | starting | mysql-partition-e97dae91e44681e0b543 | 192.0.2.60 |
| mysql-partition-e97dae91e44681e0b543/1 | failing  | mysql-partition-e97dae91e44681e0b543 | 192.0.2.61 |
| mysql-partition-e97dae91e44681e0b543/2 | failing  | mysql-partition-e97dae91e44681e0b543 | 192.0.2.62 |
+-----+-----+-----+-----+
```

Do not proceed to the next step until all three VMs are in the `starting` / `failing` state.

Update the BOSH configuration

In standard deployment, BOSH is configured to manage the cluster in a specific manner. You must change that configuration in order for the bootstrap errand to perform its work. Follow this process to make it possible for the bootstrap errand to succeed.

- i. Log into the BOSH director
- ii. Target the correct deployment
- iii. `bosh edit deployment`
 - Search for the jobs section: `jobs`
 - Search for the mysql-partition: `mysql-partition`
 - Search for the update section: `update`
 - Change `max_in_flight` to `3`.
 - Below the `max_in_flight` line, add a line: `canaries: 0`
- iv. `bosh deploy`

Run the bootstrap errand

- i. `bosh run errand bootstrap`
- ii. Validate that the errand completes successfully.
 - Some instances may still appear as `failing`. It's OK to proceed to the next step.

Restore the BOSH configuration

- i. `bosh edit deployment`
- ii. Re-set `canaries` to 1, `max_in_flight` to 1, and `serial` to true in the same manner as above.
- iii. `bosh deploy`
- iv. Validate that all mysql instances are in `running` state.

Note: It is critical that you run all of the steps. If you do not re-set the values in the BOSH manifest, the status of the jobs will not be reported correctly and can lead to troubles in future deploys.

Manual Bootstrap

1. If the bootstrap errand is not able to automatically recover the cluster, you may need to perform the steps manually. The following steps are prone to user-error and can result in lost data if followed incorrectly. Please follow the [assisted bootstrap](#) instructions above first, and only resort to the manual process if the errand fails to repair the cluster.
 - a. SSH to each node in the cluster and, as root, shut down the `mariadb` process. To SSH into BOSH-deployed VMs, see the [Advanced Troubleshooting with the BOSH CLI](#) topic.

```
$ monit stop mariadb_ctrl
```

Re-bootstrapping the cluster will not be successful unless all other nodes have been shut down.

- b. Choose a node to bootstrap.

Find the node with the highest transaction sequence number (seqno). The sequence number of a stopped node can be retained by either reading the node's state file under `/var/vcap/store/mysql/grastate.dat`, or by running a `mysqld` command with a WSREP flag, like `mysqld --wsrep-recover`.

If a node shutdown gracefully, the seqno should be in the galera state file.

```
$ cat /var/vcap/store/mysql/grastate.dat | grep 'seqno:'
```

If the node crashed or was killed, the seqno in the galera state file should be `-1`. In this case, the seqno may be recoverable from the database. The following command will cause the database to start up, log the recovered sequence number, and then exit.

```
$ /var/vcap/packages/mariadb/bin/mysqld --wsrep-recover
```

Note: The galera state file will still say `seqno: -1` afterward.


Scan the error log for the recovered sequence number (the last number after the group id (uuid) is the recovered seqno):

```
$ grep "Recovered position" /var/vcap/sys/log/mysql/mysql.err.log | tail -1
150225 18:09:42 mysqld_safe WSREP: Recovered position e93955c7-b797-11e4-9faa-9a6f0b73eb46:15
```

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's nothing to recover. Unless all nodes crashed this way, don't choose this node for bootstrapping.

Bootstrap the first node

Use the node with the highest `seqno` value as the new bootstrap node. If all nodes have the same `seqno`, you can choose any node as the new bootstrap node.

 **Note:** Only perform these bootstrap commands on the node with the highest seqno. Otherwise the node with the highest seqno will be unable to join the new cluster (unless its data is abandoned). Its mariadb process will exit with an error. See [cluster behavior](#) for more details on intentionally abandoning data.

- c. On the new bootstrap node, update state file and restart the mariadb process:

```
$ echo -n "NEEDS_BOOTSTRAP" > /var/vcap/store/mysql/state.txt
$ monit start mariadb_ctrl
```

You can check that the mariadb process has started successfully by running:

```
$ watch monit summary
```

It can take up to 10 minutes for monit to start the mariadb process.

Restart the remaining nodes

- d. After the bootstrapped node is running, start the mariadb process on the remaining nodes via monit.
Start the mariadb process:

```
$ monit start mariadb_ctrl
```

If the node is prevented from starting by the Interruptor, use the [manual steps](#) to force an SST.

- e. Verify that the new nodes have successfully joined the cluster. The following command outputs the total number of nodes in the cluster:

```
mysql> SHOW STATUS LIKE 'wsrep_cluster_size';
```

Backing Up MySQL for Pivotal Cloud Foundry

This topic describes how to enable, configure, and use backups in MySQL for Pivotal Cloud Foundry (PCF).

Overview

Automated backups have the following features:

- Periodically create and upload backup artifacts suitable for restoring the complete set of database instances allocated in the service
- No locks, no downtime
- The only effect on the serving systems is the amount of I/O required to copy the database and log files off of the VM
- Includes a metadata file that contains the critical details of the backup artifact, including the effective calendar time of the backup
- Backup artifacts are encrypted within the MySQL for PCF cluster of VMs; unencrypted data is never transported outside of the MySQL for PCF deployment

Enable Automated Backups

You can configure MySQL for PCF to automatically back up its databases to external storage.

- **How and Where:** There are two options for how automated backups transfer backup data and where the data saves out to:
 - MySQL for PCF runs an `scp` command that secure-copies backup files to a VM or physical machine operating outside of PCF. The operator provisions the backup machine separately from their PCF installation. This is the most efficient option.
 - MySQL for PCF runs an `S3` client that saves backups to an Amazon S3 bucket, `Ceph` storage cluster, or other S3-compatible endpoint certified by Pivotal.
- **When:** Backups follow a schedule that you specify with `acron` expression.
- **What:** You can back up just the primary node, or all nodes in the cluster.

To enable automated backups and configure them for options above, perform the following steps:

1. Navigate to the MySQL for Pivotal Cloud Foundry tile on the Ops Manager Installation Dashboard.
2. Click **Backups**.
3. Under **Backups**, click **Enable Backups**.

Automated Database Backups

Backups*

☐ Disable Backups (Must also choose "No Backups" below, and set Backup Prepare Node Instances to 0 in the "Resource Config" pane at left.)

☒ Enable Backups (Must also choose a destination below.)

Cron Schedule (See <http://godoc.org/github.com/robfig/cron>) *

☒ Back up all nodes

4. For **Cron Schedule**, enter a cron schedule for the backups. The syntax is similar to traditional cron, with additional features such as `@every 1d`, which specifies daily backups. See the cron Go library [documentation](#) for more information.
5. If you want to back up all nodes, select the **Back up all nodes** checkbox.
6. To enable backups using `Ceph` or AWS, continue to the [Ceph or AWS](#) section. To enable backups using SCP, continue to the [SCP](#) section.

Ceph or AWS

To back up your database on Ceph or Amazon Web Services (AWS) S3, perform the following steps:

1. Select **Ceph or Amazon S3**.

Backup Destination*

☐ No Backups

☒ Ceph or Amazon S3

S3 Endpoint URL

S3 Bucket Name *

Bucket Path *

AWS Access Key ID *

AWS Secret Access Key *

Secret


☐ SCP to a Remote Host

Save

2. Enter your **S3 Endpoint URL**. For instance, `https://s3.amazonaws.com`.

3. Enter your **S3 Bucket Name**. Do not include an `s3:// prefix`, a trailing `/`, or underscores. If the bucket does not already exist, it will be created automatically.

4. For **Bucket Path**, specify a folder within the bucket to hold your MySQL backups. Do not include a trailing `/`. If the folder does not already exist, it will be created automatically.

 **Note:** You must use this folder exclusively for this cluster's backup artifacts. Mixing the backup artifacts from different clusters within a single folder can cause confusion and possible inadvertent loss of backup artifacts.

5. For **AWS Access Key ID** and **AWS Secret Access Key**, enter your Ceph or AWS credentials. For AWS, Pivotal recommends creating an [IAM](#) credential that only has access to this bucket.
6. Click **Save**.

SCP

To back up your database using SCP, perform the following steps:

Backup Destination *

☐ No Backups

☐ Ceph or Amazon S3

☒ SCP to a Remote Host

Username *

Hostname *

Destination Directory *

Private Key *

SCP Port

22

Save

1. Select **SCP to a Remote Host**.
2. Enter the **Username**, **Hostname**, and **Destination Directory** for the backups.

 **Note:** Pivotal recommends using a VM not within the PCF deployment for the destination of SCP backups. SCP enables the operator to use any desired storage solution on the destination VM.

3. For **Private Key**, paste in the private key that will be used to encrypt the SCP transfer.
4. Enter the **SCP Port**. SCP runs on port 22 by default.
5. Click **Save**.

Disable Automated Backups

To disable automated backups, perform the following steps:

1. Navigate to the MySQL for Pivotal Cloud Foundry tile on the Ops Manager Installation Dashboard.

Automated Database Backups

Backups*

☒ Disable Backups (Must also choose "No Backups" below, and set Backup Prepare Node Instances to 0 in the "Resource Config" pane at left.)

☐ Enable Backups (Must also choose a destination below.)

Backup Destination*

☒ No Backups

☐ Ceph or Amazon S3

☐ SCP to a Remote Host

Save

- Click **Backups**.
- Under **Backups**, click **Disable Backups**.
- Under **Backup Destination**, click **No Backups**.
- Click **Save**.
- In the left navigation, click **Resource Config**.
- Change the number of instances for **Backup Prepare Node** from to .
- Click **Save**.
- Return to the Ops Manager Installation Dashboard and click **Apply Changes**.

To configure automated backups for MySQL for PCF, perform the following steps:

- Navigate to the MySQL for Pivotal Cloud Foundry tile on the Ops Manager Installation Dashboard.
- Click **Backups**.

Understand Backups

The sections below describe the [process](#) that MySQL for PCF component jobs follow when performing automated backups, and the format for the [metadata file](#) that records information about each backup.

Backup Process

Operators use Ops Manager to [configure](#) the schedule for automated backups and the location and credentials needed to store backup artifacts.

The diagram below shows the process through which MySQL for PCF jobs initiate and run automated backups.

```
sequenceDiagram
    participant Blob store
    participant Service Backup job
    Note over Service Backup job: Triggered by timer, following schedule configured in Ops Manager
    Service Backup job->>Streaming Backup client:Request backup
    Streaming Backup client->>Streaming Backup tool:Request backup
    Streaming Backup tool->>MySQL server:Request backup
    Note over MySQL server: Flush tables with read lock
    MySQL server->>Streaming Backup tool:Data Streaming Backup tool->>Streaming Backup client:Data Streaming Backup client->>Service Backup job:Data
    Note over Service Backup job: Compress and encrypt
    Service Backup job->>Blob store:Backup artifact
    Note over Blob store:Store backup artifact, using creds configured in Ops Manager
    Blob store-->>Service Backup job:Confirm artifact stored
    Note over Service Backup job:Clean up local storage
```

Two MySQL for PCF component VMs host the jobs listed above as follows:

Job	Job name in the <code></code>	Host VM
-----	-------------------------------	---------

Service Backup	<code>service-backup</code>	Backup Prepare VM
Streaming Backup client	<code>streaming-backup-client</code>	
Streaming Backup tool	<code>streaming-backup-tool</code>	MySQL VM
MySQL server	<code>mysql</code>	

Backup Metadata

Along with each backup artifact, MySQL for PCF uploads a `mysql-backup-XXXXXXXXXX.txt` metadata file.

The contents of the metadata file resemble the following:

```
compact = N
encrypted = N
tool_version = 2.4.5
server_version = 10.1.20-MariaDB
end_time = 2017-05-05 23:26:19
binlog_pos = filename 'mysql-bin.000016', position '7000000', GTID of the last change '0-1-30000'
incremental = N
format = tar
compressed = N
uuid = 30000000-3000-1000-9000-40000000000f
name =
lock_time = 0
innodb_from_lsn = 0
innodb_to_lsn = 6286393
partial = N
tool_command = --user=admin --password=... --stream=tar tmp/
ibbackup_version = 2.4.5
tool_name = innobackupex
start_time = 2017-05-05 23:26:17
```

Within this file, the most important items are the `start_time` and the `server_version` entries. Transactions that have not been completed at the start of the backup effort are not present in the restored artifact.

Note: Both `compressed` and `encrypted` show as `N` in this file, yet the artifact uploaded by MySQL for PCF is both compressed and encrypted. This is a known bug.

Restore a Backup Artifact

MySQL for PCF keeps at least two complete copies of the data. In most cases, if a cluster is still able to connect to persistent storage, you can restore a cluster to health using the [bootstrap process](#). Before resorting to a database restore, contact [Pivotal Support](#) to ensure your existing cluster is beyond help.

The disaster recovery backups feature of MySQL for PCF is primarily intended as a way to recover data to the same PCF deployment from which the data was backed up. This process replaces 100% of the data and state of a running MySQL for PCF cluster. This is especially relevant with regard to service instances and bindings.

Note: Because of how services instances are defined, you cannot restore a MySQL for PCF database to a different PCF deployment.

In the event of a total cluster loss, the process to restore a backup artifact to a MySQL for PCF cluster is entirely manual. Perform the following steps to use the offsite backups to restore your cluster to its previous state:

1. Discover the encryption keys in the **Credentials** tab of the MySQL for PCF tile.
2. If necessary, install the same version of the **MySQL for PCF** product in the Ops Manager Installation Dashboard.
3. Perform the following steps to reduce the size of the MySQL for PCF cluster to a single node:
 - a. From the Ops Manager Installation Dashboard, click the **MySQL for PCF** tile.
 - b. Click **Resource Config**.
 - c. Set the number of instances for **MySQL Server** to 1.
 - d. Click **Save**.
 - e. Return to the Ops Manager Installation Dashboard and click **Apply Changes**.

4. After the deployment finishes, perform the following steps to prepare the first node for restoration:

- SSH into the Ops Manager Director. For more information, see the [SSH into Ops Manager](#) section in the *Advanced Troubleshooting with the BOSH CLI* topic.
- Retrieve the IP address for the MySQL server by navigating to the **MySQL for PCF** tile and clicking the **Status** tab.
- Retrieve the VM credentials for the MySQL server by navigating to the **MySQL for PCF** tile and clicking the **Credentials** tab.
- From the Ops Manager Director VM, use the BOSH CLI to SSH into the first MySQL job. For more information, see the [BOSH SSH](#) section in the *Advanced Troubleshooting with the BOSH CLI* topic.
- On the MySQL server VM, become super user:

```
$ sudo su
```

- Pause the local database server:

```
$ monit stop all
```

- Confirm that all jobs are listed as `not monitored`:

```
$ watch monit summary
```

- Delete the existing MySQL data that is stored on disk:

```
$ rm -rf /var/vcap/store/mysql/*
```

5. Perform the following steps to restore the backup:

- Move the compressed backup file to the node using `scp`.
- Decrypt and expand the file using `gpg`, sending the output to tar:

```
$ gpg --decrypt mysql-backup.tar.gpg | tar -C /var/vcap/store/mysql -xvf -
```

- Change the owner of the data directory, because MySQL expects the data directory to be owned by a particular user:

```
$ chown -R vcap:vcap /var/vcap/store/mysql
```

- Start all services with `monit`:

```
$ monit start all
```

- Watch the summary until all jobs are listed as `running`:

```
$ watch monit summary
```

- Exit out of the MySQL node.

6. Perform the following steps to increase the size of the cluster back to three:

- From the Ops Manager Installation Dashboard, click the **MySQL for PCF** tile.
- Click **Resource Config**.
- Set the number of instances for **MySQL Server** to `3`.
- Click **Save**.
- Return to the Ops Manager Installation Dashboard and click **Apply Changes**.

Perform Manual Backup

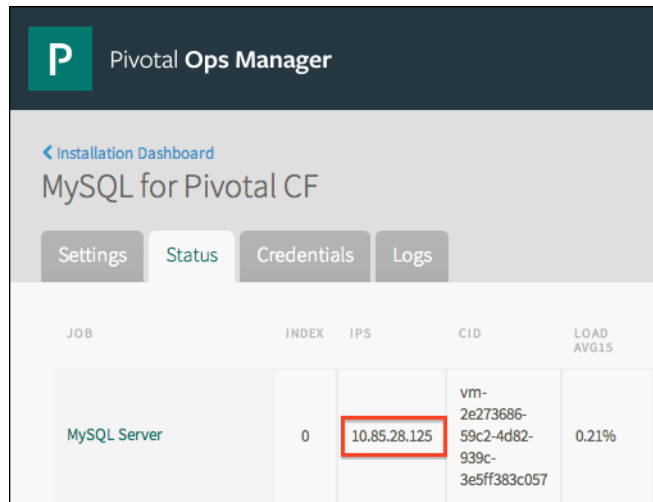
If you do not want to use the automated backups included in MySQL for PCF, you can perform backups manually.

Retrieve IP Address and Credentials

Perform the following steps to retrieve the IP address and credentials required for a manual backup:

- From the Ops Manager Installation Dashboard, click the **MySQL for PCF** tile.

2. Click the **Status** tab.



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

3. Locate the IP address for the MySQL node under **MySQL Server**.
4. In the **Credentials** tab, from the **MySQL Server** job and **Mysql Admin Password** name, obtain the admin password.

Manual Backup

Back up your data manually with [mysqldump](#). This backup acquires a global read lock on all tables, but does not hold it for the entire duration of the dump.

To back up all databases in the MySQL deployment:

```
$ mysqldump -u admin -p -h $MYSQL_NODE_IP --all-databases --single-transaction > user_databases.sql
```

To back up a single database, specify the database name:

```
$ mysqldump -u admin -p -h $MYSQL_NODE_IP $DB_NAME --single-transaction > user_databases.sql
```

Manual Restore

The procedure for restoring from a backup is the same whether one or multiple databases were backed up. Executing the SQL dump will drop, recreate, and refill the specified databases and tables.

WARNING: Restoring a database deletes all data that existed in the database before the restore. Restoring a database using a full backup artifact, produced by `mysqldump --all-databases` for example, replaces all data and user permissions.

Prepare to restore:

- If running in HA configuration, reduce the size of the MySQL for PCF cluster to a single node, following the [restore instructions](#) above.
- Locate the MySQL Admin credentials in the **Credentials** tab, as above.
- Use the MySQL password and IP address to enable the creation of tables using any storage engine.

```
$ mysql -u admin -p -h $MYSQL_NODE_IP -e "SET GLOBAL enforce_storage_engine=NULL"
```

- Use the MySQL password and IP address to restore the MySQL databases by running the following command.

```
$ mysql -u admin -p -h $MYSQL_NODE_IP < user_databases.sql
```

- Use the MySQL password and IP address to restore original storage engine restriction.

```
$ mysql -u admin -p -h $MYSQL_NODE_IP -e "SET GLOBAL enforce_storage_engine='InnoDB'"
```


- To restore HA mode, re-configure MySQL for PCF to run using three nodes in the same way as the [restoring instructions](#) above.
- If not running HA mode, it's important to restart the database server. This step is not necessary if scaling back to three MySQL nodes.


```
$ monit stop mariadb_ctrl  
$ monit start mariadb_ctrl
```

For more examples of manual backup and restore procedures, see the [MariaDB documentation](#) .

Scaling Down MySQL

This topic describes how to safely scale down your MySQL for Pivotal Cloud Foundry (PCF) cluster to a single node.

By default MySQL for PCF is a single node. To take advantage of the high availability features of MySQL for PCF, you may have scaled the configuration up to three nodes.

 **Note:** If you are only running the MySQL cluster with a single node, you do not need to perform these steps.

Check the Health of Your Cluster

Before scaling down your MySQL cluster, perform the following actions to ensure the cluster is healthy.

1. Obtain the IP addresses of your MySQL server by performing the following steps:
 - a. From the Pivotal Cloud Foundry (PCF) **Installation Dashboard**, click the **MySQL for Pivotal Cloud Foundry** tile.
 - b. Click the **Status** tab.
 - c. Record the IP addresses for all instances of the **MySQL Server** job.
2. Obtain the admin credentials for your MySQL server by performing the following steps:
 - a. From the MySQL for PCF tile, click the **Credentials** tab.
 - b. Locate the **Mysql Admin Password** entry in the **MySQL Server** section and click **Link to Credential**.
 - c. Record the values for `identity` and `password`.
3. 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.
4. From the Ops Manager VM, place some data in the first node by performing the following steps, replacing `FIRST-NODE-IP-ADDRESS` with the IP address of the first node retrieved above and `YOUR-IDENTITY` with the `identity` value obtained above. When prompted for a password, provide the `password` value obtained above.
 - a. Create a dummy database in the first node:

```
$ mysql -h FIRST-NODE-IP-ADDRESS -u YOUR-IDENTITY -p -e "create database verify_healthy;"
```

- b. Create a dummy table in the dummy database:

```
$ mysql -h FIRST-NODE-IP-ADDRESS -u YOUR-IDENTITY -p -D verify_healthy -e "create table dummy_table (id int not null primary key auto_increment, info text) engine=InnoDB;"
```

- c. Insert some data into the dummy table:

```
$ mysql -h FIRST-NODE-IP-ADDRESS -u YOUR-IDENTITY -p -D verify_healthy -e "insert into dummy_table(info) values ('dummy data'),('more dummy data'),('even more dummy data');" 
```

- d. Query the table and verify that the three rows of dummy data exist on the first node:

```
mysql -h FIRST-NODE-IP-ADDRESS -u YOUR-IDENTITY -p -D verify_healthy -e "select * from dummy_table;"
Enter password:
+----+-----+
| id | info                |
+----+-----+
| 4  | dummy data          |
| 7  | more dummy data     |
| 10 | even more dummy data |
+----+-----+
```

5. Verify that the other nodes contain the same dummy data by performing the following steps for each of the remaining MySQL server IP addresses obtained above:

- a. Query the dummy table, replacing `NEXT-NODE-IP-ADDRESS` with the IP address of the MySQL server instance and `YOUR-IDENTITY` with the `identity` value obtained above. When prompted for a password, provide the `password` value obtained above.

```
$ mysql -h NEXT-NODE-IP-ADDRESS -u YOUR-IDENTITY -p -D verify_healthy -e "select * from dummy_table;"
```

- b. Examine the output of the `mysql` command and verify that the node contains the same three rows of dummy data as the other

nodes.

```
+---+-----+
| id | info      |
+---+-----+
| 4  | dummy data |
| 7  | more dummy data |
| 10 | even more dummy data |
+---+-----+
```

6. If each MySQL server instance does not return the same 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.

Scale Down Your Cluster

1. Delete the dummy database, replacing `FIRST-NODE-IP-ADDRESS` with the IP address of the first MySQL server node and `YOUR-IDENTITY` with the `identity` value obtained above. When prompted for a password, provide the `password` value obtained above.

```
$ mysql -h FIRST-NODE-IP-ADDRESS -u YOUR-IDENTITY -p -e "drop database verify_healthy;"
```

2. From the PCF **Installation Dashboard**, click the **MySQL for Pivotal Cloud Foundry** tile.
3. Click the **Settings** tab.
4. Click **Resource Config** and use the drop-down menu to change the **Instances** count for **MySQL Server** to `1`.
5. Click **Save** to apply the changes.

Rotating MySQL for PCF Credentials

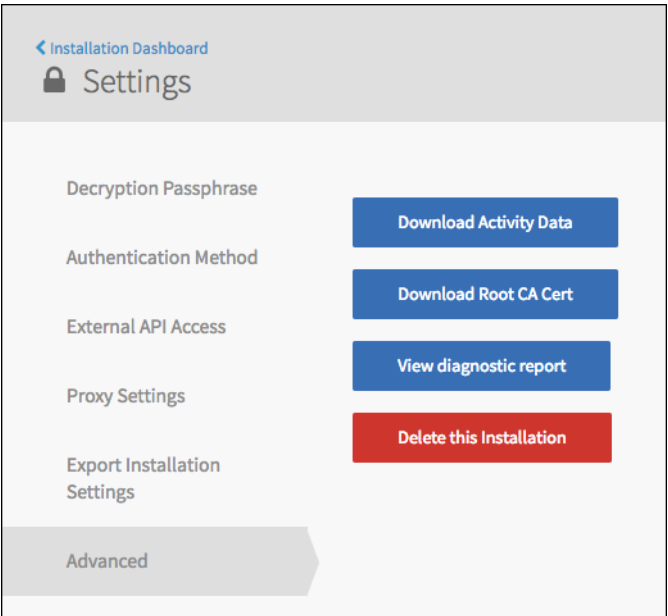
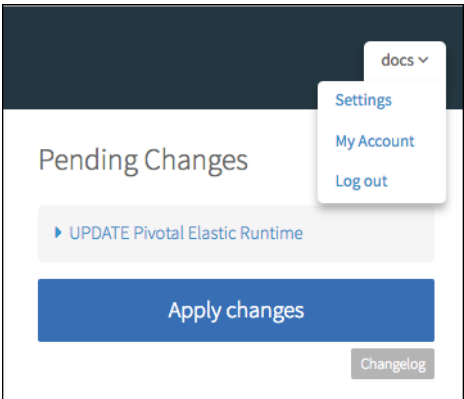
Page last updated:

This topic describes how to rotate credentials for MySQL for Pivotal Cloud Foundry (MySQL for PCF). If you are also using Elastic Runtime MySQL, review the notes in this procedure in order to rotate credentials for both products.

Prerequisites

To perform the steps below, you need to obtain the following:

- 1. Your root CA certificate in a `.cert` file. To retrieve the root CA certificate of your deployment, follow these steps:
 - a. In Ops Manager, click your username located at the top right and choose **Settings**.



- b. Click **Advanced**.
 - c. Click **Download Root CA Cert**
- 2. Your MySQL for PCF root password. To retrieve your MySQL for PCF root password, navigate to the Ops Manager Installation Dashboard and select **MySQL for Pivotal Cloud Foundry > Credentials** Your MySQL for PCF root password is called `Mysql Admin Password`.

MySQL Server	Vm Credentials	Link to Credential
	Mysql Admin Password	Link to Credential

Note: If you use Elastic Runtime MySQL, you also need your Elastic Runtime MySQL root password. To retrieve your Elastic Runtime MySQL root password, navigate to the Ops Manager **Installation Dashboard** and select **MySQL > Credentials**. Your Elastic Runtime MySQL root password is called `Mysql Admin Credentials`.

Rotate Your MySQL for PCF Credentials

1. Install the User Account and Authentication (UAA) Command Line Interface (UAAC).

```
$ gem install cf-uaac
```

2. Make sure `uaac` gem is installed.

```
$ which uaac
/Users/pivotal/.gem/ruby/2.3.0/bin/uaac
```

3. Target your Ops Manager UAA and provide the path to your root CA certificate.

```
$ uaac target https://YOUR-OPSMAN-FQDN/uaa/ --ca-cert YOUR-ROOT-CA.crt
Target: https://YOUR-OPSMAN-FQDN/uaa/
```

4. Get your token with `uaac token owner get`.
 - Enter `opsman` for `Client ID`.
 - Press enter for `Client secret` to leave it blank.
 - Use the user name and password you used above to log into the Ops Manager web interface for `User name` and `Password`.

```
$ uaac token owner get
Client ID: opsman
Client secret:
User name: admin
Password: *****
Successfully fetched token via owner password grant.
Target: https://YOUR-OPSMAN-FQDN/uaa
Context: admin, from client opsman
```

5. Run the following command to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.


```
$ uaac context
[1][https://YOUR-OPSMAN-FQDN/uaa]
skip_ssl_validation: true
ca_cert: /Users/pivotal/.ssh/YOUR-ROOT-CA.crt
[0]*[admin]
user_id: 75acfdfa-9449-4497-a093-ce40ded250ac
client_id: opsman
access_token: LONG_ACCESS_TOKEN_STRING
token_type: bearer
refresh_token: LONG_REFRESH_TOKEN_STRING
expires_in: 43199
scope: clients.read opsman.user uaa.admin scim.read opsman.admin clients.write scim.write
jti: 8419c793d377429aa40eea07fb6c7686
```

6. Create a file called `uaac-token` that contains only the `LONG_ACCESS_TOKEN_STRING` from the output above.
7. Use `curl` to make a request to the Ops Manager API. Authenticate with the contents of the `uaac-token` file and pipe the response into `installation_settings_current.json`.

```
$ curl -skH "Authorization: Bearer $(cat uaac-token)" https://YOUR-OPSMAN-FQDN/api/installation_settings > installation_settings_current.json
```

8. Check to see that the MySQL for PCF [root password](#) is in the current installation settings file:

```
$ grep -c YOUR-MYSQL-FOR-PCF-ROOT-PASSWORD installation_settings_current.json
```


 **Note:** If you use Elastic Runtime MySQL, you should also run the following command: `$ grep -c YOUR-ERT-MYSQL-ROOT-PASSWORD installation_settings_current.json`

9. Remove the root password from the installation settings file.

```
$ sed -e 's/"value":{"identity":"root","password":"[^"]*"},"("identifier":"mysql_admin")/1/g' installation_settings_current.json > installation_settings_updated.json
```

10. Validate that the root password has been removed from the `installation_settings_updated.json` file.

```
$ grep -c YOUR-MYSQL-FOR-PCF-ROOT-PASSWORD installation_settings_updated.json
0
```


 **Note:** If you use Elastic Runtime MySQL, you should also run the following command: `$ grep -c YOUR-ERT-MYSQL-ROOT-PASSWORD installation_settings_updated.json`

11. Upload the updated installation settings.

```
$ curl -skX POST -H "Authorization: Bearer $(cat uaac-token)" "https://YOUR-OPSMAN-FQDN/uaa/api/installation_settings" -F 'installation[file]=@installation_settings_updated.json' {}
```

12. Navigate to the Ops Manager **Installation Dashboard** and click **Apply Changes**.
13. Once the installation has completed, validate that the MySQL for PCF root password has been changed. Retrieve the new [password](#) from **MySQL > Credentials**. Use the IP address for the **MySQL Proxy** located in the **Status** tab.

```
$ mysql -uroot -p -h 198.51.100.1
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
[...]
```

 **Note:** If you use Elastic Runtime MySQL, you should also validate that the Elastic Runtime MySQL root password has been changed. Retrieve the new password from **Elastic Runtime > Credentials**. Use the IP address for the **MySQL Proxy**, located in the **Status** tab.

Running mysql-diag

This topic discusses how to use the `mysql-diag` tool in MySQL for Pivotal Cloud Foundry (PCF). `mysql-diag` relays the state of your MySQL service and suggests steps to take in the event of a node failure. In conjunction with Pivotal Support, this tool helps expedite the diagnosis and resolution of problems with MySQL for PCF.

In MySQL for PCF 1.9.0 and later, `mysql-diag` is automatically installed and configured. If you are running MySQL for PCF 1.8.x and earlier, then you will need to create a configuration file in order to use `mysql-diag`.

Prepare Your Environment

MySQL for PCF 1.9.0 and later ships with the `mysql-diag` tool and comes with an automatically generated configuration file. In versions 1.9.0. and later, you can find `mysql-diag` on the `mysql-monitor` node. If you are running MySQL for PCF 1.8.x or earlier then you must download `mysql-diag` and create a configuration file. If you do not have a monitor node, as is the case with some older versions of the software, Pivotal recommends that you use one of the mysql cluster nodes instead.

Only complete the download and configuration instructions below if you are on MySQL for PCF 1.8.x or earlier.

Download and Run mysql-diag

To download `mysql-diag`:

1. Download the file labeled **mysql-diag.conf** attached to the [Diagnosing problems with Elastic Runtime MySQL or the Pivotal MySQL Tile](#) Knowledge Base article.
2. Copy that binary to the `mysql-monitor` VM with the following command: `bosh scp JOB-NAME JOB-INSTANCE-NUMBER --upload LOCAL-FILE-PATH REMOTE-FILE-PATH`

Running the `bosh instances` command will display the information needed to insert the **JOB-NAME** and **JOB-INSTANCE-NUMBER** options. For more information on the `bosh instances` command, see the [bosh documentation](#) on system administration tasks. The **LOCAL-FILE-PATH** option is the path to where you want to locate the **mysql-diag.conf** file. The **REMOTE-FILE-PATH** option is the initial location of the **mysql-diag.conf** file.

1. Execute the **mysql-diag.conf** file with the following command:

```
mysql-diag -c ./mysql-diag.conf
```

Configure mysql-diag

To configure `mysql-diag`:

1. Paste the Configuration File Template below into a text editor

```
{
  "mysql": {
    "username": "repcanary",
    "password": "password",
    "port": 3306,
    "nodes": [
      {
        "host": "10.244.7.4",
      },
      {
        "host": "10.244.8.4",
      },
      {
        "host": "10.244.9.4",
      }
    ]
  }
}
```

2. Replace the passwords with the values that you find in **OpsMan** within the **Credentials** tab.

3. Copy the completed template into the same VM that you downloaded the `mysql-diag` tool, using the `bosh scp` command.
4. Move the configuration file to the same directory as the `mysql-diag` tool.
5. Run the following command in order to start the tool:

```
$ mysql-diag -c ./diag-config.json
```

mysql-diag-agent

MySQL for PCF 1.9.0 and later will have the `mysql-diag-agent` present. Versions 1.8.x and earlier of MySQL for PCF do not have the `mysql-diag-agent`. If the `mysql-diag-agent` is not available, your output from the `mysql-diag` tool will not include the percentage of Persistent and Ephemeral Disk space used by a Host.

Example Healthy Output

Upon running `mysql-diag` in your terminal, you will see the following if your canary status is healthy:

```
Checking canary status...healthy
```

Here is a sample `mysql-diag` output after the tool has identified a healthy cluster in a MySQL for PCF version that does not contain the `mysql-diag-agent`:

```
Checking cluster status of mysql/a1 at 10.0.16.44 ...
Checking cluster status of mysql/c3 at 10.0.32.10 ...
Checking cluster status of mysql/b2 at 10.0.16.45 ...
Checking cluster status of mysql/a1 at 10.0.16.44 ... done
Checking cluster status of mysql/c3 at 10.0.32.10 ... done
Checking cluster status of mysql/b2 at 10.0.16.45 ... done
+-----+-----+-----+-----+-----+
| HOST   | NAME/UUID | WSREP LOCAL STATE | WSREP CLUSTER STATUS | WSREP CLUSTER SIZE |
+-----+-----+-----+-----+-----+
| 10.0.16.44 | mysql/a1 | Synced           | Primary               | 3 |
| 10.0.32.10 | mysql/c3 | Synced           | Primary               | 3 |
| 10.0.16.45 | mysql/b2 | Synced           | Primary               | 3 |
+-----+-----+-----+-----+-----+
I don't think bootstrap is necessary
Checking disk status of mysql/a1 at 10.0.16.44 ...
Checking disk status of mysql/c3 at 10.0.32.10 ...
Checking disk status of mysql/b2 at 10.0.16.45 ...
Checking disk status of mysql/a1 at 10.0.16.44 ... dial tcp 10.0.16.44: getsockopt: connection refuse
Checking disk status of mysql/c3 at 10.0.32.10 ... dial tcp 10.0.16.44: getsockopt: connection refuse
Checking disk status of mysql/b2 at 10.0.16.45 ... dial tcp 10.0.16.44: getsockopt: connection refuse
```

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 cluster, running `mysql-diag` outputs actionable steps meant to expedite the recovery of the cluster. Below is a sample `mysql-diag` output after the tool identified an unhealthy cluster in a MySQL for PCF version that does not contain the `mysql-diag-agent`:


```

Checking cluster status of mysql/a1 at 10.0.16.44 ...
Checking cluster status of mysql/c3 at 10.0.32.10 ...
Checking cluster status of mysql/b2 at 10.0.16.45 ...
Checking cluster status of mysql/a1 at 10.0.16.44 ... dial tcp 10.0.16.44: getsockopt: connection refused
Checking cluster status of mysql/c3 at 10.0.32.10 ... dial tcp 10.0.32.10: getsockopt: connection refused
Checking cluster status of mysql/b2 at 10.0.16.45 ... dial tcp 10.0.16.45: getsockopt: connection refused

+-----+-----+-----+-----+-----+
|  HOST   | NAME/UUID | WSREP LOCAL STATE | WSREP CLUSTER STATUS | WSREP CLUSTER SIZE |
+-----+-----+-----+-----+-----+
| 10.0.16.44 | mysql/a1 | N/A - ERROR      | N/A - ERROR          | N/A - ERROR        |
| 10.0.16.45 | mysql/b2 | N/A - ERROR      | N/A - ERROR          | N/A - ERROR        |
| 10.0.32.10 | mysql/c3 | N/A - ERROR      | N/A - ERROR          | N/A - ERROR        |
+-----+-----+-----+-----+-----+

Checking disk status of mysql/a1 at 10.0.16.44 ...
Checking disk status of mysql/c3 at 10.0.32.10 ...
Checking disk status of mysql/b2 at 10.0.16.45 ...
Checking disk status of mysql/a1 at 10.0.16.44 ... dial tcp 10.0.16.44: getsockopt: connection refused
Checking disk status of mysql/c3 at 10.0.32.10 ... dial tcp 10.0.32.10: getsockopt: connection refused
Checking disk status of mysql/b2 at 10.0.16.45 ... dial tcp 10.0.16.45: getsockopt: connection refused

[CRITICAL] The replication process is unhealthy. Writes are disabled.

[CRITICAL] Run the download-logs command:
$ download-logs -d /tmp/output -n 10.0.16.44 -n 10.16.45 -n 10.0.32.10
For full information about how to download and use the download-logs command see https://discuss.pivotal.io/hc/en-us/articles/221504408

[WARNING]
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 “bosh recreate” and “bosh cck”. These options remove logs on the VMs making it harder to diagnose cluster issues.

```