Pivotal

PRODUCT DOCUMENTATION

MySQL for PCF®

Version 1.9

User's Guide

Table of Contents

MySQL for PCF

Note: MySQL v1.9 is no longer supported. The support period for v1.9 has expired. To stay up-to-date with the latest software and security updates, upgrade to MySQL v1.10 or later.

This is documentation for the MySQL for Pivotal Cloud Foundry (PCF) service tile. The tile can be downloaded from Pivotal Network.

This documents MySQL for PCF v1.9, a pre-provisioned service that supports single-node or high-availability cluster architectures. For the on-demand service that creates dedicated service instances dynamically, see the v2.2 documentation.

About MySQL

MySQL is a powerful open-source relational database used by apps since the mid-90s. Developers have relied on MySQL as a first step to storing, processing and sharing data. As its user community has grown, MySQL has become a robust system capable of handling a wide variety of use cases and very significant workloads. Unlike other traditional databases which centralize and consolidate data, MySQL lends itself to dedicated deployment supporting the “shared nothing” context of building apps in the cloud.

About MySQL for PCF

The MySQL for PCF product delivers a fully managed, “Database as a Service” to Cloud Foundry users. The tile deploys and maintains a MySQL server running a recent release of MariaDB and Galera; an SQL proxy, Switchboard; and a service broker. The tile configures sane defaults for a general-use relational database service.

With MySQL for PCF installed, developers can attach a database to their apps in as little as two commands, `cf create-service` and `cf bind-service`. Developers can retrieve connection credentials in the standard manner, from the `VCAP_SERVICES` environment variable. Developers can select from a menu of service plans options, which are configured by the platform operator.

You can deploy MySQL for PCF in either single-node or high availability (HA) topology. The HA topology has three MySQL servers, two proxies, and two service brokers, and you need to supply a load balancer.

Product Snapshot

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

Element	Details
Version	v1.9.18
Release date	February 7, 2018
Software component version	MariaDB v10.1.26, Galera v25.3.20
Compatible Ops Manager version(s)	v1.9.x and v1.10.x
Compatible Elastic Runtime version(s)	v1.9.21+ and v1.10.9+; **not v1.11.x**
IaaS support	AWS, GCP, Azure, OpenStack, and vSphere
IPsec support	Yes

MySQL for PCF and Other PCF Services

Some PCF services offer *on-demand* service plans. These plans let developers provision service instances when they want.

These contrast with the more common *pre-provisioned* service plans, which require operators to provision the service instances during installation and configuration through the service tile UI.

The following PCF services offer on-demand service plans:

- MySQL for PCF v2.0 and later

- RabbitMQ for PCF
- Redis for PCF
- Pivotal Cloud Cache (PCC)

These services package and deliver their on-demand service offerings differently. For example, some services, like Redis for PCF, have one tile, and you configure the tile differently depending on whether you want on-demand service plans or pre-provisioned service plans.

For other services, like PCC, you install one tile for on-demand service plans and a different tile for pre-provisioned service plans.

The following table lists and contrasts the different ways that PCF services package on-demand and pre-provisioned service offerings.

PCF service tile	Standalone product related to the service	Versions supporting on demand	Versions supporting pre-provisioned
RabbitMQ for PCF	Pivotal RabbitMQ	v1.8 and later	All versions
Redis for PCF	Redis	v1.8 and later	All versions
MySQL for PCF	MySQL	v2.x (based on Percona Server)	v1.x (based on MariaDB and Galera)
PCC	Pivotal GemFire	All versions	*NA*
GemFire for PCF	Pivotal GemFire	*NA*	All versions

Prepare to Upgrade

Before upgrading to the latest version of MySQL for PCF, do the following:

1. Confirm that you are on Ops Manager v1.9 or later and are on MySQL for PCF v1.8 or later.

Ops Manager Version	Supported Upgrades from Imported MySQL Installation	
	From	To
v1.9.x, v1.10.x	v1.8.x	latest v1.9.x
	v1.9.x	
v1.11.x	v1.8.x	Not supported
	v1.9.x	Not supported
		latest v1.10.x

   For more information about upgrade versions, see the Product Compatibility Matrix.

2. If you are running in the HA configuration, check the health of the cluster:

   a. Download, configure, and run the mysql-diag tool.
   b. If `mysql-diag` shows any cluster issues, fix them.
   c. Do not apply changes to the upgrade until the issues have been resolved.

3. As of v1.9.0, MySQL for PCF enables strict mode. This means that where previously large indices were silently truncated, now `ALTER TABLE` and `CREATE INDEX` statements may fail with an error.

   IMPORTANT: If any of your apps already use or want to create indices with more than 767 bytes, then create or alter the apps to use `ROW_FORMAT DYNAMIC` or `ROW_FORMAT COMPRESSED`.

   A workaround for failures due to strict mode is to use the `IGNORE` keyword: this converts the error to a warning.
   For more information about these changes, see MySQL Server Tuning and Defaults.

Table Locks are Now Configurable

In previous releases, Pivotal announced the intent to deprecate table locks for all service bindings. However, Pivotal now leaves this choice up to the operator, for more information, see Allow Table Locks.

Enterprise Readiness by Topology

The table below shows the enterprise-readiness of each MySQL for PCF topology. Consult the Known Issues topic for information about issues in current releases of MySQL for PCF.

	Single-Node	High Availability (HA)
MySQL	1 node	3-node cluster
SQL Proxy	1 node	2 nodes
Service Broker	1 node	2 nodes
High Availability	*n/a*	Yes
Multi-AZ Support	*n/a*	Yes *
Rolling Upgrades	*n/a*	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	*n/a*	*n/a*
Long-lived Canaries	*n/a*	Replication Canary

(*) vSphere and AWS (1.8.0 and later)
(†) Requires IPSEC BOSH plug-in

High Availability Limitations

When deployed in HA topology, MySQL for PCF runs three master nodes. This cluster arrangement imposes some limitations that you should be aware of, which do not apply to single-node MySQL database servers.

- Although two proxy instances are deployed by default, there is no automation to direct clients from one to the other. See the note in the Proxy section. To address this, configure a load balancer as described in the Proxy section.
- MySQL for PCF only supports the InnoDB storage engine; it is the default storage engine for new tables. Pre-existing tables that are not InnoDB are at risk because they are not replicated within a cluster.
- The database servers are shared, managed by multi-tenant processes to serve apps across the PCF deployment. Although data is securely isolated between tenants using unique credentials, app performance may be impacted by noisy neighbors.
- Round-trip latency between database nodes must be less than five seconds. Latency exceeding this results in a network partition. If more than half of the nodes are partitioned, the cluster loses quorum and become unusable until manually bootstrapped.
- See also the list of Known Limitations in MariaDB cluster.

Release Notes

Consult the Release Notes for information about changes between versions of this product.

Known Issues

Consult the Known Issues topic for information about issues in current releases of MySQL for PCF.

Feedback

Please provide any bugs, feature requests, or questions to the Pivotal Cloud Foundry Feedback list.

Release Notes

Important: This is the last planned release of MySQL for PCF v1.9. The support period for v1.9 has expired. To stay up-to-date with the latest software and security updates, upgrade to MySQL for PCF v1.10 or later.

Note: MySQL for PCF v1.9 is certified to work with PCF v1.9 and v1.10. For PCF v1.11, upgrade to MySQL for PCF v1.10.

v1.9.18

Release Date: February 7, 2018

- **Bug fix:** `mysql-diag` now fails quickly when VMs in the cluster are not available.
- **Bug fix:** Replication canary causes high memory usage spikes because it overwrites rather than appends to logs files.

v1.9.17

Release Date: January 24, 2018

- Update to stemcell 3421.38 to address:
  - USN-3540-2

v1.9.16

Release Date: January 19, 2018

- Update to stemcell 3421.37 to address:
  - USN-3522-2
  - USN-3522-4
  - USN-3534-1

  For more information, see pivotal.io/security.

v1.9.15

Release Date: December 13, 2017

- Update to stemcell 3421.34 to address CVE-2017-1000405.
  For more information, see pivotal.io/security.

v1.9.14

Release Date: December 8, 2017

MySQL for PCF v1.9.14 resolves several bugs, updates dependencies to maintain security, and includes changes that improve syslog and metrics.

- **Syslog Format Change**
  - The components of MySQL for PCF, MySQL, Proxy, MySQL, Backups and Monitoring have been configured to observe RFC 5424 message format when sending logs to syslog.
- **New Disk Usage Metrics**
  - MySQL for PCF now produces disk space metrics and how much space has been reserved for service instances. For more information, see Persistent

and Ephemeral Disk Free and Service Plans Allocated in the KPI documentation.

- **Dependency Updates**
  - Updated to stemcell 3421.32 to address low and medium vulnerabilities.
  - Updated versions of Ruby and associated libraries, and socat to avoid known vulnerabilities.
- **Bug Fixes**
  - **Bug fix:** Fixed a rare bug which left hanging backup processes on the MySQL nodes if a client disconnects unexpectedly.
  - **Bug fix:** streaming-mysql-backup-tool was running as `root`, now it runs user `vcap`.
  - **Bug fix:** Fixed a rare bug in which the replication canary could write to a node which was not a member of the cluster, causing the cluster to be in an inconsistent state.
- **IPsec**
  - **Bug fix:** MySQL failed to start if the node once had IPsec installed but IPsec is no longer installed.
  - **Bug fix:** IPsec caused mariadb_ctrl to be left in an `Execution Failed` state when taking time to start.

v1.9.13

Release Date: November 3, 2017

- **New:** Changed the default of `innodb_flush_log_at_trx_commit` to `2`. This is considered safe when using Galera in clustered mode. Single node and very conservative cluster deployments might want to reset to the former default, `1`. For details, see InnoDB Flush Log policy: 2.
- Update to MariaDB 10.1.26
- Upgrade nokogiri to 1.8.1, and rubygems to 2.6.13, to address recent Common Vulnerabilities and Exposures (CVEs)
- Update to stemcell 3421.31 to address low and medium stemcell vulnerabilities.
- **Bug fix:** Smoke tests do not run if there are no public plans.

v1.9.12

Release Date: September 29, 2017

- Provides a checkbox where the operator can configure table locking behavior. For more information, see the allow table locks section of the configuration page.
- Bumped the default size of the backup VMs from 1 GB RAM / 2 CPUs to 4 GB RAM / 4 CPUs. You can customize this default in the Resource Configuration pane.
- **Bug fix:** Pre-start phase logs warnings, rather than time out, when `mysqld` takes time to start or stop.
- **Security:** Updates to rubygems 2.6.6 to address CVE-2017-0902.

v1.9.11

Release Date: September 21, 2017

- Updates stemcell to 3363.37. This security upgrade resolves USN-3420-2.
  For more information, see pivotal.io/security.

v1.9.10

Release Date: September 14, 2017

- Re-names package dependencies to avoid installation conflicts.
- Updates stemcell to 3363.35, which addresses a minor logrotate issue.

v1.9.9

Release Date: August 21, 2017

- Updates stemcell to 3363.31. This security upgrade resolves USN-3392-2.
  For more information, see pivotal.io/security.

v1.9.8

Release Date: August 16, 2017

- Updates stemcell to 3363.30. This security upgrade resolves USN-3385-2
  For more information, see pivotal.io/security.

v1.9.7

Release Date: August 11, 2017

- **Change the Interruptor default setting to OFF.**
  For a year, MySQL for PCF has included the Interruptor. It's a protective mechanism that stops a node from automatically rejoining the cluster if doing so might 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 want 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.**
- Small improvements to the `mysql-diag` tool, now including a warning when available disk space is low.
- 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`
- Updates stemcell to 3312.32. This security upgrade resolves USN-3265-2.
  For more information, see pivotal.io/security.

v1.9.6

Release Date: July 28, 2017

- MySQL for PCF v1.9 has been updated with several security and performance tuning enhancements. See the architecture and configuration documentation for more details. Some important highlights:
  - **Note**: Changes default to restrict administrator access to localhost only. If you access the `admin` or `roadmin` accounts from outside of the server VMs, you must configure this option to allow access.
  - Adds two features which prevent clients from accessing the server file system.
  - Tuning features of InnoDB buffer pool size, strict mode, flush method and large prefix settings.
- The URL for the proxy has changed. The old scheme for each proxy dashboard was:

  ```
  proxy-JOB-INDEX-p-mysql.SYSTEM-DOMAIN
  ```

  For example: `proxy-0-p-mysql.sys.bosh-lite.com`
  The new scheme is now:

  ```
  JOB-INDEX-proxy-p-mysql.SYSTEM-DOMAIN
  ```

  For example: `0-proxy-p-mysql.sys.bosh-lite.com`
- There is a new Advanced Configuration to set a cluster's name before initial deployment.
- MariaDB has been updated to v10.1.24 and Galera to v25.3.20.
- Updated nokogiri to address CVE-2017-5029.

v1.9.5

Release Date: June 22, 2017

- There is a 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.

  To address this, MySQL for PCF now has a separate configuration, similar to that of 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.
- MySQL for PCF v1.9 has new backup storage options. Backup artifacts can now be stored on Google Cloud Storage or Azure Blob Storage.
- Updated stemcell to 3312.29. This security upgrade resolves USN-3334-1.
  For more information, see pivotal.io/security.

v1.9.4

Release Date: June 2, 2017

- Updated stemcell to 3312.28. This security upgrade resolves USN-3291-3.
  For more information, see pivotal.io/security.

v1.9.3

Release Date: May 19, 2017

- MySQL for Pivotal Cloud Foundry (PCF) now emits performance metrics which can be read via the Loggregator Firehose. This allows operators to monitor and understand performance aspects of the MySQL cluster. For more information, see Monitoring the MySQL Service.
- 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 may be due to an issue where the default probe timeout is too long while running under IPsec, and should be reduced. Version 1.9.3 of MySQL for PCF allows you to reduce the New Cluster Probe Timeout in the MySQL server configuration page. For more information, see MySQL Server Configuration.
  - **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.

v1.9.2

Release Date: April 27, 2017

- Changed Proxy API route hostname from `proxy-0-p-mysql` to `proxy-api-p-mysql`.
- Updated stemcell to 3312.24. This security upgrade resolves USN-3265-2.
- **Bug fix:** Addressed an issue where backups were unable to store backups on AWS S3 regions that require the v4 signature.
- **Bug fix:** Addressed an issue where nodes may fail to rejoin the cluster after restart. See the Rejoin Unsafe Fails Known Issue for more details.
- Note: The title of the tile now appears as “MySQL for PCF,” not simply “MySQL.”

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

v1.9.1

Introducing mysql-diag

If configured, the monitoring VM comes with a pre-configured `mysql-diag` tool. This tool gives you a snapshot of the current state of the cluster.

Pivotal recommends that you always run `mysql-diag` before upgrade. Attempting to upgrade while a cluster is in a less-than-healthy state will not go smoothly. Always be sure that the cluster is healthy before upgrading.

The `mysql-diag` tool also works on earlier releases of MySQL for PCF, but it won't be pre-installed and configured. Instructions and a download are located in the Pivotal Knowledgebase.

MySQL Server Tuning and Defaults

- Several configuration settings have been moved to a new configuration pane, **MySQL Server Configuration**
- sql_mode is now set: `NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION,STRICT_ALL_TABLES`. `sql_mode` was not set on previously versions.
  - `NO_ENGINE_SUBSTITUTION`: Guards against applications unintentionally creating non-InnoDB tables. Previously, the current master would accept, but not replicate such tables across the cluster.
  - `NO_AUTO_CREATE_USER`: Prevents implicitly creating users without a password.
  - `STRICT_ALL_TABLES`: Enables strict mode
  - **Warning:** Enabling strict mode means that Applications may now start seeing errors, such as when inserting strings that are too long, or numeric values that are out of range. Previously, MariaDB silently modified the data on commit.
  - Additionally, these settings no longer allow a user to specify large indices despite `innodb_large_index=ON` without additionally specifying `ROW_FORMAT=DYNAMIC` in each table create statement. Apps that do not specify this additional clause will see an error during DDL similar to: `ERROR 1709 (HY000): Index column size too large. The maximum column size is 767 bytes.`
    - The IGNORE keyword can be used when strict mode is set to convert the error to a warning.
- wsrep_max_ws_size is set to the maximum, 2GB
- wsrep_max_ws_rows enforces the default of 0
  - For more information about these limitations, see the documentation on transaction size.
  - There is a known issue in which this setting also affects DDLs. It will be fixed in a future release of MariaDB.
- There is a known issue that sometimes causes the `rejoin-unsafe` errand to fail. See Rejoin Unsafe Fails Known Issue for more information.

v1.9.0

Version 1.9.0 was not released. Use v1.9.1.

Architecture

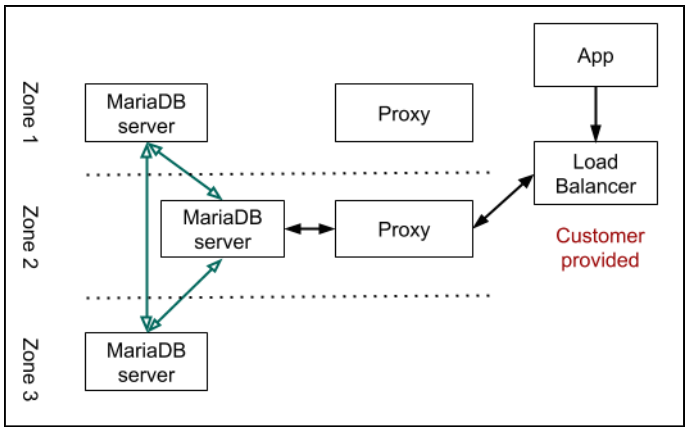
This topic describes the high-level component architecture of the MySQL for Pivotal Cloud Foundry (PCF) service, in both its single-node and HA configurations. It also describes the defaults and other design decisions applied to the service's MariaDB servers and the Galera Cluster that manages them.

HA Topology

In HA topology, the MySQL for PCF service uses the following:

- Three MySQL servers running MariaDB as the MySQL engine and Galera to keep them in sync
- Two Switchboard proxies that direct queries to a single backend and provide superfast failover in the event of a backend server failure
- A customer-provided load balancer that directs traffic to one of the proxies, or to the other proxy in the event of proxy failure
- Two service brokers that create and bind service instances with failover in the event of a service broker failure

If you use multiple availability zones, the MySQL installer spreads MySQL and Proxy components across zones to make the service durable in the event of a zone failure.



How It Works

At any time, MySQL for PCF normally only sends queries to one of the three MySQL servers. The server that receives the queries is the*current master* node. Galera ensures that the other two servers mirror any changes made on the current master.

In a three-node Galera cluster, two nodes define a quorum that automatically maintains availability when any node becomes inaccessible. This happens when the host fails or in the event of a network partition.

When a single node cannot communicate with the rest of the cluster, it stops accepting traffic. The other two nodes continue to function normally. When the isolated node regains connectivity, it rejoins the cluster.

If two nodes are no longer able to connect to the cluster, quorum is lost and the cluster becomes inaccessible. To bring the cluster back up, you must restart it manually, as documented in the bootstrapping topic.

Avoid an Even Number of Nodes

Pivotal recommends deploying either one or three MySQL server nodes, avoiding an even number. This prevents a network partition from causing the entire cluster to lose quorum, because neither side has more than half of the nodes.

The minimum number of nodes required to tolerate a single node failure is three. With a two-node cluster, a single node failure causes loss of quorum.

Cluster Scaling Behavior

This section explains how the MySQL cluster behaves when you change its number of nodes.

Scaling Up from One to Three Nodes

When you add new MariaDB nodes to an existing node, they replicate data from the existing primary node and join the cluster once replication is complete. Performance may temporarily degrade while the new nodes sync all data from the original node.

After new nodes have joined the cluster, the proxy continues to route incoming connections to the primary node.

If the proxy detects that the primary node is unhealthy, it severs connections to the primary node and routes all new connections to a different, healthy node.

If the proxy detects that no healthy MariaDB nodes exist in the cluster, it rejects all connections and the cluster becomes inaccessible.

How Apps Are Affected by Scaling Up

When you scale MySQL for PCF from a single-node to a multi-node high availability topology, this single node is restarted and reconfigured as a part of a cluster. During the restart, apps consuming MySQL for PCF lose their connection to the node. The apps should be able to reconnect after the MySQL node is restarted.

Scaling Down from Three to One Node

When you scale down from multiple MariaDB nodes to a single node, the primary node continues to survive as the sole remaining node (provided it remains healthy), and the proxy continues to route incoming connections to the node.

Graceful Removal of Nodes

Removing nodes from a cluster *gracefully* means that the cluster size decreases while the cluster maintains its healthy state.

You can remove nodes gracefully by manually shutting each down with `monit` or by decreasing the cluster size as described in the Switch Between Single and HA Topologies section of the Configuring MySQL for PCF topic.

MySQL Server Defaults for MariaDB Components

This section describes the defaults that the MySQL for PCF tile applies to its MariaDB components. There are additional properties which can be customized. See the configuration documentation for more information.

Max User Connections

To ensure all users get fair access to system resources, MySQL for PCF defaults each user's number of connections to 40. Operators can override this setting for each service plan in the **Service Plans** configuration pane.

Skip External Locking

Each virtual machine only runs one `mysqld` process, so they do not need external locking.

Max Allowed Packet: 256 MB

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

Query Cache Type and Size

MySQL for PCF matches MariaDB's default as of version 10.1.7 and later, by disabling `query_cache_type` and setting `query_cache_size` to 0.

InnoDB File-Per-Table Tablespaces

InnoDB allows using either a single file to represent all data, or a separate file for each table. MySQL for PCF uses a separate file for each table (`innodb_file_per_table = ON`) to optimize flexibility and efficiency. For a full list of pros and cons, see MySQL's documentation for InnoDB File-Per-Table Mode.

InnoDB File Format: Barracuda

MySQL for PCF uses the `Barracuda` file format to take advantage of extra features available with the `innodb_file_per_table = ON` option.

InnoDB Log Files: 1 GB

MySQL for PCF clusters default to a log-file size of 1 GB to support large blobs.

InnoDB Buffer Pool Size

Read the MariaDB documentation on the InnoDB Buffer Pool for more information.

InnoDB Flush Method

Defines the method used to flush data to InnoDB data files, which can affect I/O throughput. MySQL for PCF uses MariaDB's default, `fsync()`. Read the MariaDB documentation on InnoDB Flush Method for more information.

InnoDB Flush Log Policy: 2

Galera performs replication before writing to disk. If a single node crashes before the buffer is flushed, the data is safe because Galera has already replicated the transaction to other nodes in the cluster. If all nodes crash simultaneously, it is possible to lose up to a second of transactions.

- MySQL for PCF v1.9.13 and v1.10.6 set innodb_flush_log_at_trx_commit to `2`, which improves the performance of heavily active clusters. Both `0` and `2` are considered safe when running a Galera cluster, because you can recover writes from another node. The last second's transactions are not flushed to disk **only** if all nodes in the cluster crash simultaneously.
- When set to `2`, the buffer is flushed at commit but it is not synced. This means that you can safely kill `mysqld`, but if the OS or the host crashes, you might lose some transactions on the local node.
- For full ACID compliance, set this policy to `1`, which flushes to disk after every commit. Previous versions of MySQL for PCF v1 used `1`.
- When set to `0`, the buffer is not flushed at each commit. This means that the buffer can be lost if the `mysqld` process crashes. If you run `kill -9 mysqld`, you might lose some transactions on the local node.

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 that will use large index prefixes.

See also the section on SQL mode before disabling.

InnoDB Strict Mode

Enabling this feature causes the database storage engine (InnoDB) to return errors instead of warnings in certain cases. For more information, refer to the MariaDB strict mode documentation.

Temporary Tables

MySQL converts temporary in-memory tables to temporary on-disk tables when either of the following occurs:

- A query generates more than 16 million rows of output
- A query uses more than 32 MB of data space

Read the MariaDB documentation on tmp_table_size for more information.

Table Open Cache

Read the MariaDB documentation on optimizing table_open_cache size for more information.

Table Definition Cache Size

Read the MariaDB documentation on table_definition_cache for more information.

Reverse Name Resolution Off

Disabling reverse DNS lookups improves performance. Clearing this option causes the MySQL servers to perform a reverse DNS lookup on each new connection. Typically, MySQL uses this to restrict access by hostname, but MySQL for PCF uses user credentials to authenticate access, not hostnames. Thus, most deployments do not need reverse DNS lookups.

Slow Query Log

MySQL for PCF automatically enables the slow query log 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.

sql_mode Additionally Enables Strict Mode

sql_mode is set to: `NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION,STRICT_ALL_TABLES`. `sql_mode` was not set on versions before p-mysql v1.8.0.

- `NO_ENGINE_SUBSTITUTION`: Guards against apps unintentionally creating non-InnoDB tables. Previously, the current master would accept, but not replicate such tables across the cluster.
- `NO_AUTO_CREATE_USER`: Prevents implicitly creating users without a password.
- `STRICT_ALL_TABLES`: Enables strict mode
  - Strict mode means that apps may see errors, such as when inserting strings that are too long, or numeric values that are out of range. Without this setting the default MariaDB behavior is to silently modify data on commit.
  - Strict mode also does not allow a user to specify large indices even if `innodb_large_index` is set to `ON`. Addtionally, users need to specify `ROW_FORMAT=DYNAMIC` or `ROW_FORMAT=COMPRESSED` in each table create statement. Apps that do not specify this additional clause see an error during DDL similar to: `ERROR 1709 (HY000): Index column size too large. The maximum column size is 767 bytes.`
  - The IGNORE keyword can be used when strict mode is set to convert the error to a warning.

Security Configuration Defaults

Restrict Client Access to Server File System

The MariaDB setting secure_file_priv is set to only allow file access from the server directory, `/var/vcap/data/mysql/files/`. This prevents clients from reading arbitrary files on the server file system. If a user wishes to do a bulk-import, an administrator must first deliver the file to that directory.

Skip Symbolic Links

MySQL for PCF is configured to prevent the use of symlinks to tables. This is a recommended security setting which prevents users from manipulating files on the server's file system.

Allow Client to Send Files

This option allows clients to send files to the server. Disabling ensures that clients will not be directed to inadvertently share data with unknown database servers.

Allow Remote Admin Access

When enabled, `mysql` clients are allowed to connect as administrator from remote hosts. When disabled, administrators must `bosh ssh` into each MySQL VM to connect as the MySQL super user. This is a server setting; network configuration and Application Security Groups restrictions may still limit a client's ability to establish a connection with the databases.

This setting affects the admin and roadmin accounts.

Note: Some errands, notably `verify-cluster-schemas`, require administrator access. Some errands fail when remote admin access is disabled.

Allow Command History

When disabled, this prohibits the creation of command line history files on the MySQL nodes. If administrators frequently `ssh` into the MySQL VMs without using `bosh ssh`, the default setting of Enabled automatically creates history files of their interactions with the server. These files may contain sensitive information. Disable this option to ensure that no trace files are left on the file system.

MySQL Server Defaults for Galera Components

This section describes some defaults that the MySQL for PCF tile applies to its Galera components.

SST Method: Xtrabackup

When a new node is added to or rejoins a cluster, a primary node from the cluster is designated as the state donor. Then the new node synchronizes with the donor through the State Snapshot Transfer (SST) process.

MySQL for PCF uses Xtrabackup for SST, which lets the state donor node continue accepting reads and writes during the transfer. Galera defaults to using `rsync` to perform SST, which is usually fastest, but blocks requests during the transfer.

Large Data File Splitting Enabled

MySQL for PCF enables `wsrep_load_data_splitting` to split large data imports into separate transactions. This facilitates loading large files into a MariaDB cluster.

Maximum Transaction Sizes

These are the maximum transaction sizes set for Galera:

- wsrep_max_ws_size is set to 2 GB, the maximum allowed by Galera.
- wsrep_max_ws_rows enforces the default of 0. Specifying no limit avoids a known issue with MariaDB.

For general information, see Transaction Size.

Known Issues

This topic lists known issues with specific releases of MySQL for Pivotal Cloud Foundry (PCF).

No metrics on PCF v1.11

All v1.9.x releases of MySQL for PCF do not produce metrics when installed on PCF v1.11 or later.

Install fails when using IPsec

In releases previous to MySQL for PCF v1.9.3, when IPsec is also installed, the product may fail to deploy. This may be due to an issue where the default probe timeout is too long while running under IPsec, and should be reduced. Version 1.9.3 of MySQL for PCF allows you to reduce the New Cluster Probe Timeout in the MySQL server configuration page. For more information, see MySQL Server Configuration.

Rejoin Unsafe Fails

In releases previous to MySQL for PCF v1.9.2, due to a timeout in the startup procedure, if a node takes a long time to catch up while rejoining the cluster, the procedure appears to fail. This has been fixed in v1.9.2.

Node Cannot Rejoin

In releases previous to MySQL for PCF v1.9.2, there is a condition in which, if MariaDB crashes, and further fails during re-start, the startup processes does not notice the failure. Most frequently, this happens when MariaDB legitimately crashes during normal operation, and when restarting, needs to SST but is prevented by the Interruptor. Be aware that there are other conditions when a node crashes on startup (such as full disk, persistent storage issue, etc). To resolve this, it is necessary to run `monit stop mariadb_ctrl` on the failing node, before you can successfully run the `rejoin-unsafe` errand.

This failure condition can happen silently. Therefore, you must monitor the state of your cluster by watching `wsrep_cluster_size`, watching the proxy page, or via `mysql-diag` tool, to notice that a node is unable to restart. This has been fixed in v1.9.2.

Installing MySQL for PCF

This topic explains how to install MySQL for Pivotal Cloud Foundry (PCF).

Plan your Deployment

Network Layout

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

To optimize uptime, deploy a load balancer in front of the SQL Proxy nodes. Configure the 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. This eliminates the first proxy instance as a single point of failure. See Configure a Load Balancer below for load balancer configuration recommendations.

If you deploy the MySQL service on a different network than Elastic Runtime, configure the firewall rules as follows to allow traffic between Elastic Runtime and the MySQL service.

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

Configure a Load Balancer

For high availability, Pivotal recommends using a load balancer in front of the proxies:

- **Configure your load balancer for failover-only mode.** Failover-only mode sends all traffic to one proxy instance at a time, and redirects to the other proxy only if the first proxy fails. This behavior prevents deadlocks when different proxies send queries to update the same database row. This can happen during brief server node failures, when the active server node changes. Amazon ELB does not support this mode; see AWS Route 53 for the alternative configuration.
- **Make your idle time out long enough to not interrupt long-running queries.** When queries take a long time, the load balancer can time out and interrupt the 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.
- **Configure a healthcheck or monitor, using TCP against port 1936**. This defaults to TCP port `1936`, to maintain backwards compatibility with previous releases. This port is not configurable. Unauthenticated healthchecks against port 3306 may cause the service to become unavailable and require manual intervention to fix.
- **Configure the load balancer to route traffic for TCP port 3306 to the IPs of all proxy instances on TCP port 3306.**

After you install MySQL for PCF, you assign IPs to the proxy instances in Ops Manager.

Add a Load Balancer to an Existing Installation

If you initially deploy MySQL for PCF v1.5.0 without a load balancer and without proxy IPs configured, you can set up a load balancer later to remove the proxy as a single point of failure. When adding a load balancer to an existing installation, you need to:

- Rebind your apps to receive the hostname or IP that resolves to the load balancer. To rebind: 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 Ops Manager, configure DNS for your load balancer to point to the IPs that were dynamically assigned to your proxies. You can find these IPs in the **Status** tab. Configuration of proxy IPs after the product is deployed with dynamically assigned IPs is not well supported.

Create an Application Security Group

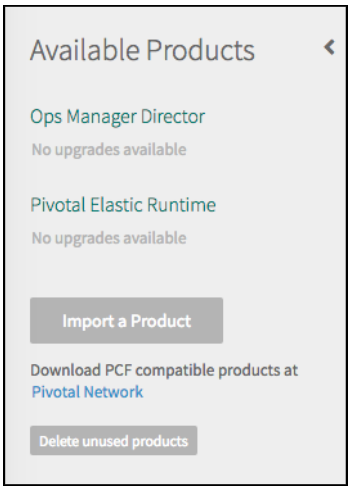
Create an Application Security Group (ASG) for MySQL for PCF. See Creating Application Security Groups for MySQL for instructions.

The ASG allows smoke tests to run when you install the MySQL for PCF service and allows apps to access the service after it is installed.

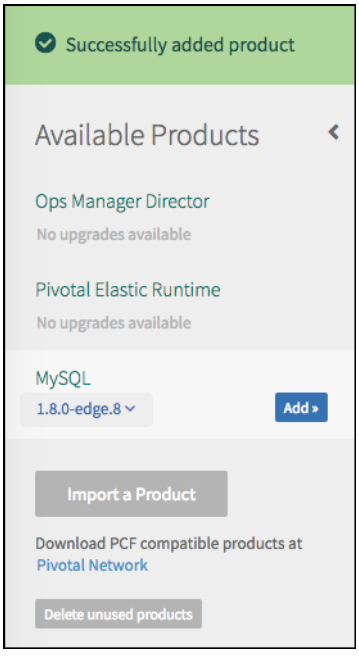
Note: The service is not installable or usable until an ASG is in place.

Install the MySQL for PCF Tile

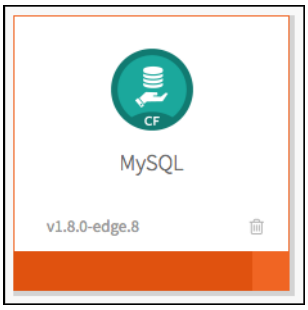
1. Download the product file from Pivotal Network.
2. Navigate to the Ops Manager Installation Dashboard.



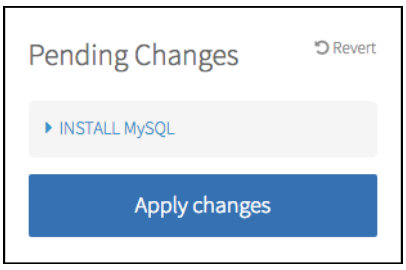
3. Click **Import a Product** to upload the product file to your Ops Manager installation.



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 configure the settings for your MySQL for PCF service, including its service plans. See Configuring MySQL for PCF for instructions.

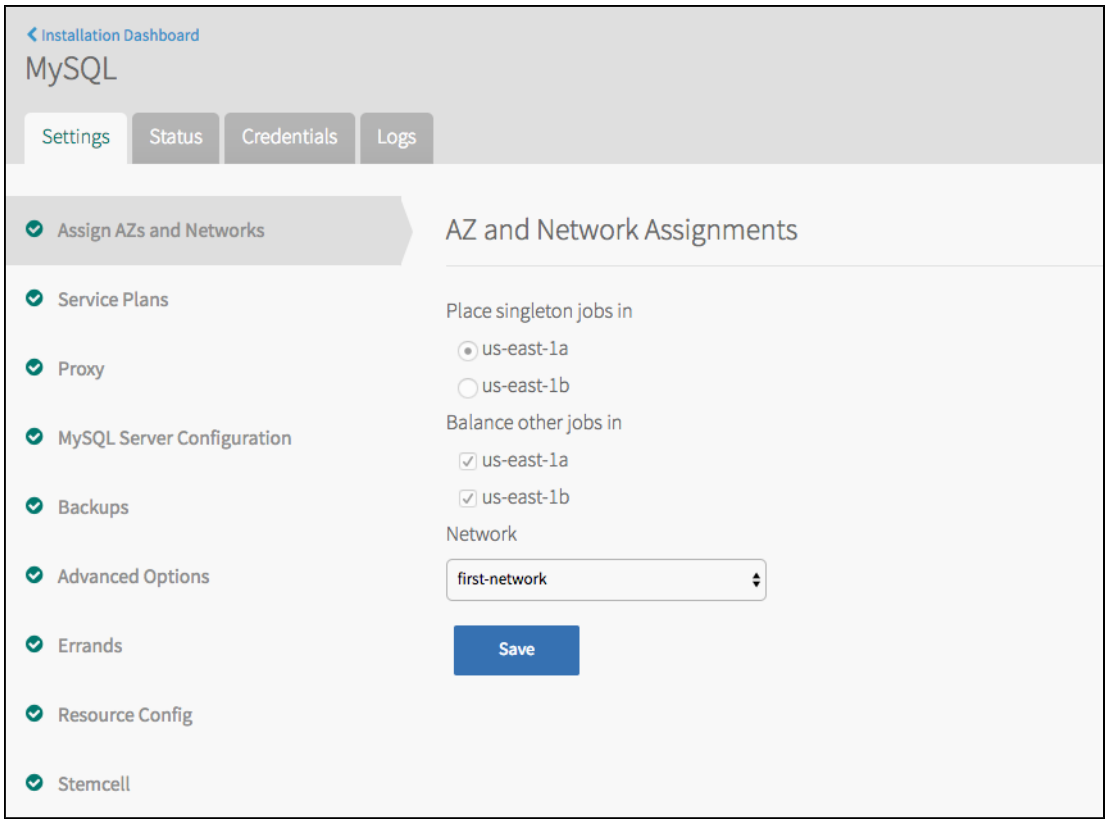


6. Click **Apply Changes** to deploy the service.

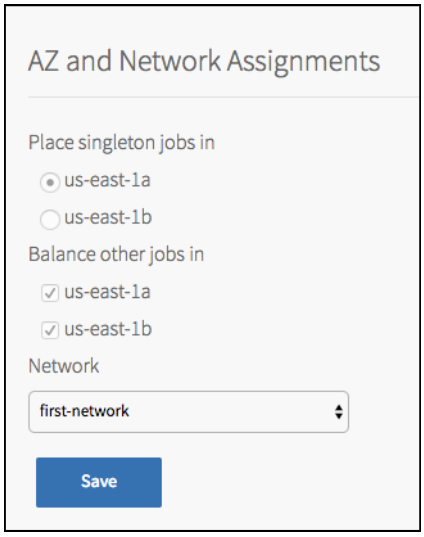
Configuring MySQL for PCF

This topic explains how to configure the MySQL for PCF service.

To configure your MySQL service, click the **MySQL for PCF** tile in the Ops Manager **Installation Dashboard**, open each pane under the **Settings** tab, and review or change the configurable settings as described in the sections below.



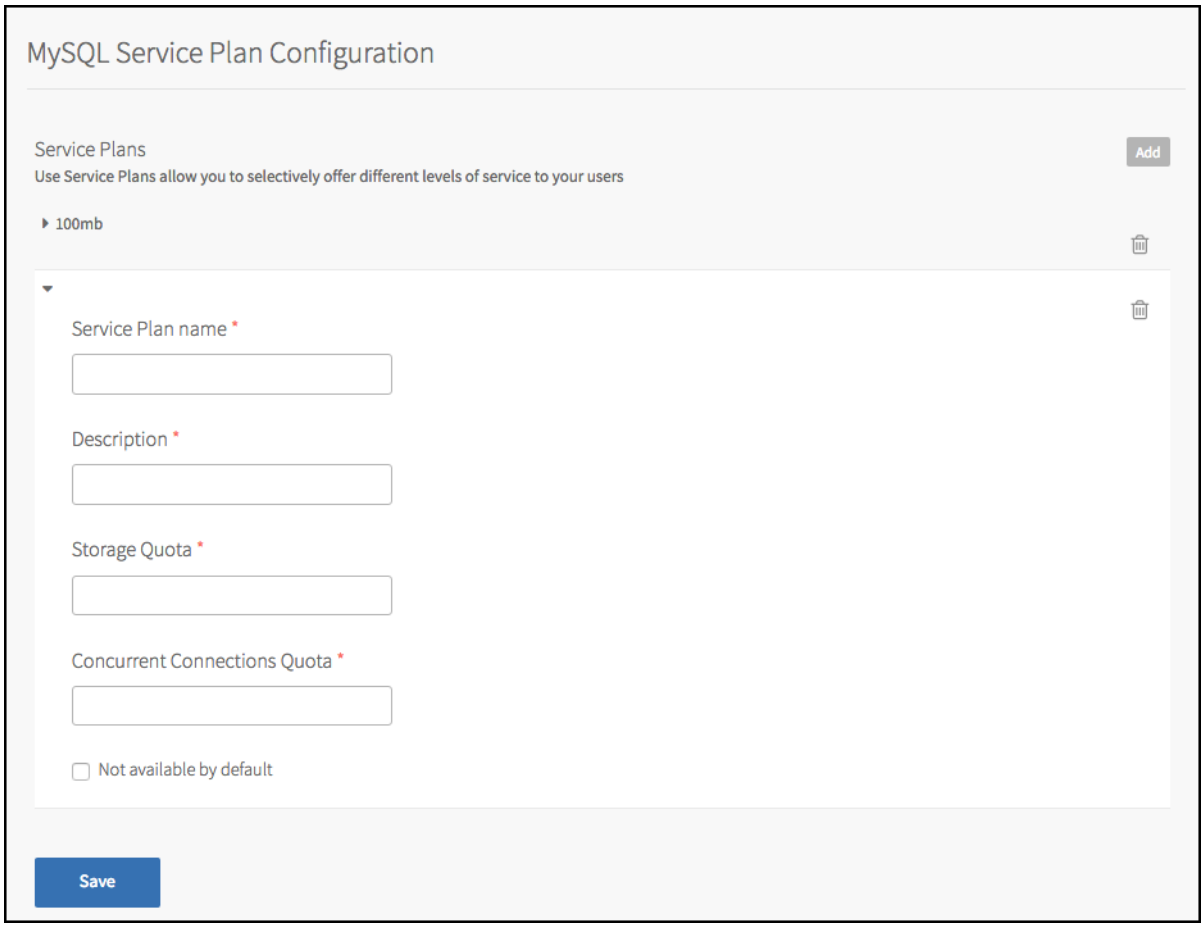
Assign AZs and Networks



MySQL for PCF supports deployment to multiple availability zones (AZs).

To maximize uptime, 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 load balancer configuration instructions for details.

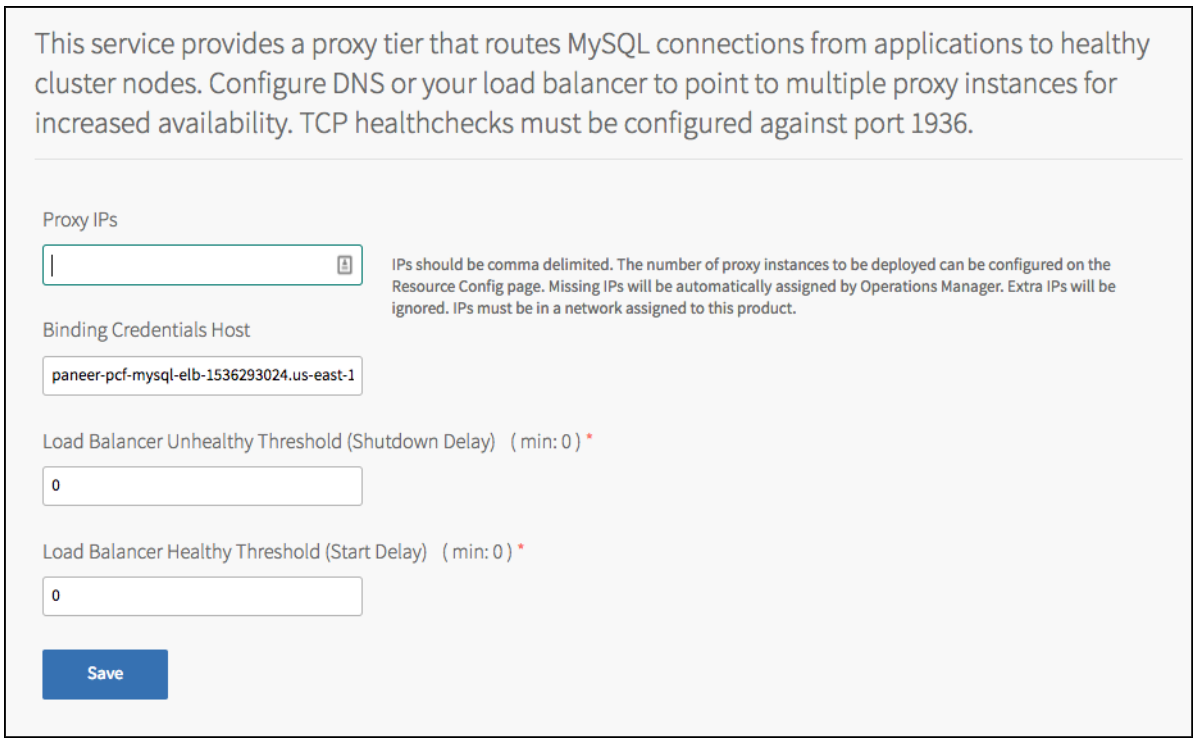
Service Plans



Service plans offer developers different versions of the MySQL service. An example is tiered service plans with that offer a range of resource limits and pricing. See Service Plans for how to configure one or more service plans in the **Service Plans** pane.

Note: You cannot deploy MySQL for PCF without at least one service plan defined.

Proxy



The proxy tier routes connections from apps to healthy MariaDB cluster nodes, even in the event of node failure.

- 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.

To enable their PCF apps to use a MySQL database, developers bind their apps to instances of the MySQL for PCF service. For more information, see Application Binding. By default, the MySQL service provides bound apps with the IP address of the first instance in the proxy tier, even with multiple proxy instances deployed. This makes the first proxy instance a single point of failure unless you deploy a load balancer.

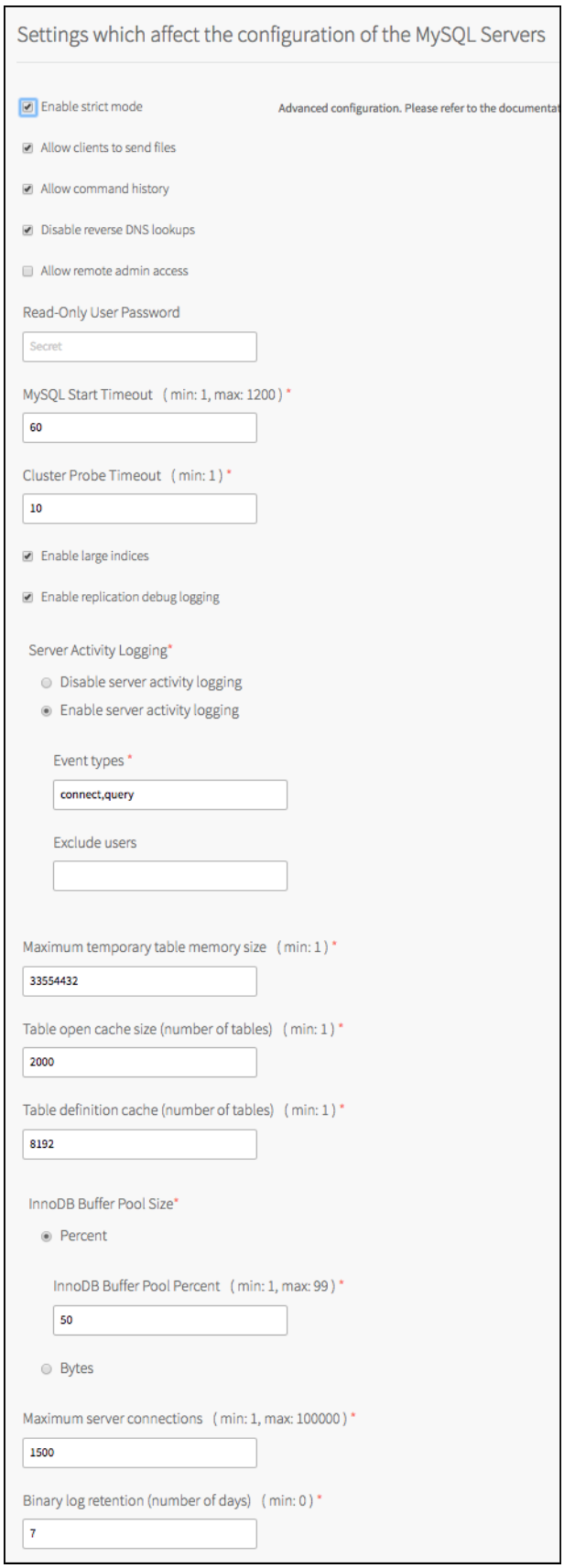
Note: To eliminate the first proxy instance as a single point of failure, configure a load balancer to route client connections to all proxy IPs, and configure the MySQL service to give bound applications a hostname or IP address that resolves to the load balancer.

Proxy Count Cannot be Reduced

Once an operator deploys MySQL for PCF, they cannot reduce the number of proxy IPs or proxy instances, and cannot remove the configured IPs from the **Proxy IPs** field.

If the product is initially deployed without proxy IPs, adding IPs to the **Proxy IPs** field can only add additional proxy instances. Scaling down is unpredictably permitted, and the first proxy instance can never be assigned an operator-configured IP.

MySQL Server Configuration



You can change the following options to suit your environment. Since these changes affect all service instances, exercise caution when making changes.

For more information about how MySQL for PCF is configured, see the architecture documentation.

- **Enable InnoDB Strict Mode**
  Default: Enabled
  See the architecture documentation for information on this setting.

- **Allow Clients to Send Files**
  Default: Enabled
  See the architecture documentation for information on this setting.
- **Allow Command History**
  Default: Enabled
  See the architecture documentation for information on this setting.
- **Disable Reverse DNS lookups**
  Default: Enabled
  See the architecture documentation for information on this setting.
- **Allow Remote Admin Access**
  Default: Disabled
  See the architecture documentation for information on how to configure this setting. Enabling this checkbox will affect how you back up and restore your MySQL data manually. See Perform Manual Backup.
- **Read-Only User Password**
  Default: Disabled
  Activates a special user, `roadmin`, a read-only administrator. Supply a special password for administrators who require the ability to view all of the data maintained by the MySQL for PCF installation. Leaving this field blank de-activates the read-only user.
- **MySQL Start Timeout**
  Default: 60 seconds
  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 failed to start properly, and appear as failing in Ops Manager and BOSH output. Depending on the data stored by the database, and the time represented in logs, you may need to increase this above the default of 60 seconds.
- **New Cluster Probe Timeout**
  Default: 10 seconds
  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. In cases of 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** might fail the deployment. To account for this, either increase the **MySQL Start Timeout** or lower the **New Cluster Probe Timeout** so that a new node doesn't spend too long performing this test.
- **Allow Table Locks**
  Default: Enabled
  When enabled, clients can acquire table locks on the node processing the transaction. This is enabled for backwards compatibility. Pivotal recommends disabling this for all new deployments, because table locks are not replicated across cluster nodes.
- **Enable Large Indices**
  Default: Enabled
  See the architecture documentation for information on this setting.
- **Enable replication debug logging**
  Default: Enabled
  Directs MySQL for PCF service to log replication error events. Disable this option only if error logging is overloading your logging systems' capacity.
- **Server Activity Logging**
  The MySQL service includes the MariaDB Audit plugin to log server activity. 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 server logging is enabled, the file is rotated every 100 megabytes, and the 30 most recent files are retained.
  - Event types
    Default: `connect,query`
  - Exclude users
    Supply a comma-separated list of additional database users that should not be included in the activity log. Note that these users are always excluded form audit logging:
    - `repcanary`
    - `mysql-metrics`
    - `cluster-health-logger`
    - `galera-healthcheck`
    - `quota-enforcer`

  Note: Due to the sensitive nature of these logs, they are not transmitted to the syslog server.
- **Maximum Temporary Table Memory Size**
  Default: 33554432 bytes (32 MB)
  The maximum size (in bytes) of internal in-memory temporary tables. See the architecture documentation for information on this setting.
- **Table Open Cache Size**

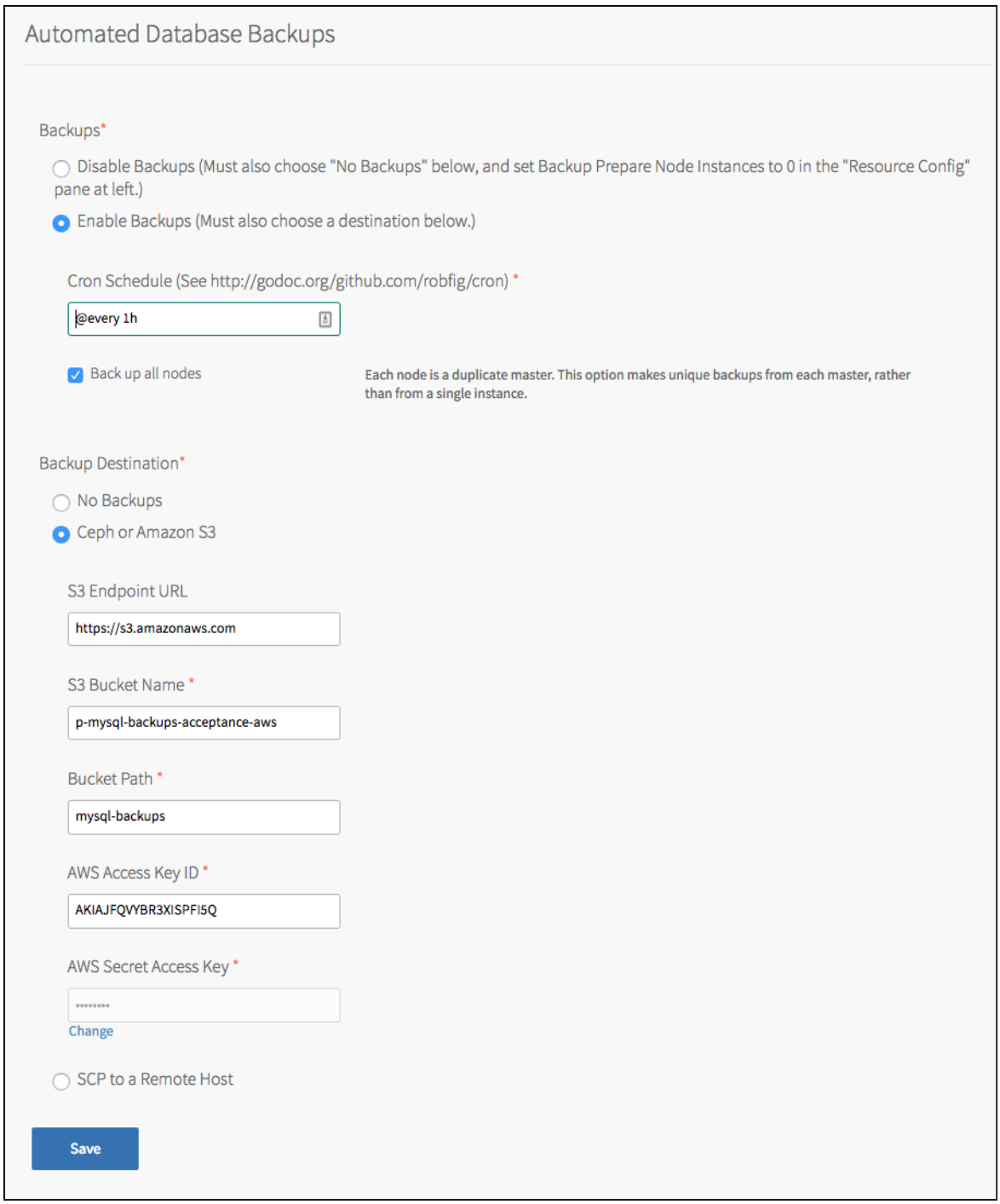
Default: 2000 tables
The number of table handles to keep open. See the architecture documentation for information on this setting.

- **Table Definition Cache Size**
  Default: 8192 tables
  Set this to a number relative to the number of tables the server will manage. See the architecture documentation for information on this setting.
- **InnoDB Buffer Pool Size**
  Default: 50 percent of instance RAM
  Choose Percent to configure the percent of system RAM allocated to the memory buffer used by InnoDB. Choose Bytes to configure the size in bytes, instead. See the architecture documentation for information on this setting.
- **InnoDB Log Flush Timing**
  Default: 2
  InnoDB log buffer is written to the log file at each commit; the log file is flushed to disk once per second. Set to `1` when running in non-HA mode. This default improves cluster performance. For more information, see the InnoDB Flush Method. The MariaDB default, `1`, is to flush the log to disk at every commit, which is required for ACID compliance.

  > **Note**: The most recent second's transactions is flushed to disk *only* if all nodes crash simultaneously. All other cluster lifecycle events do not affect data retention. To eliminate this risk, set Log Flush Timing to `1`, which improves the performance of heavily active clusters.

- **Maximum Server Connections**
  Default: 1500
  The maximum number of simultaneous client connections for the entire deployment, regardless of service instance.
- **Binary Log Retention Time**
  Default: 7 days
  Time in days to store binary logs before purging. These logs are not used by MySQL for PCF. They are stored only for diagnostic purposes.

Backups

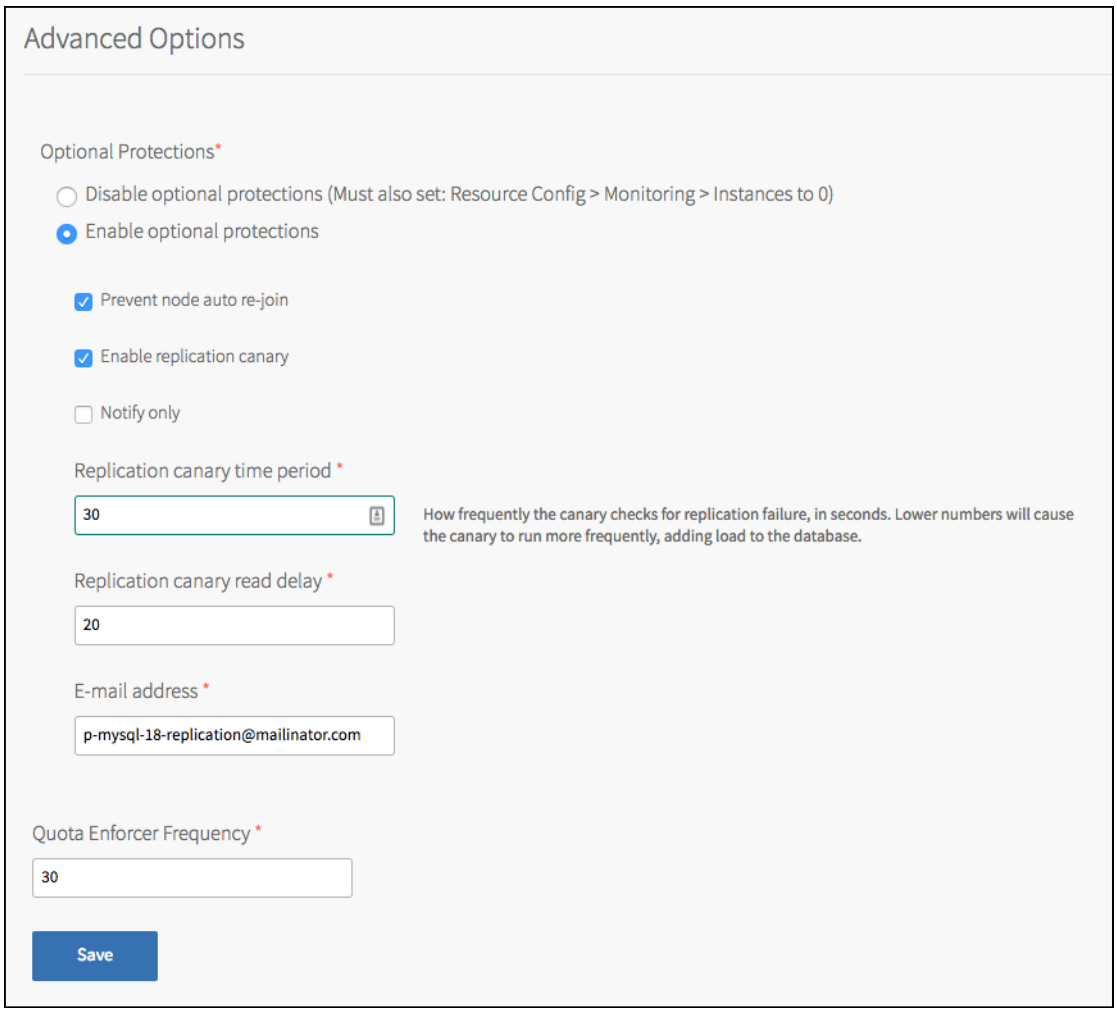


The **Backups** pane configures automated database backups. To configure backups:

1. Choose **Disable Backups** or **Enable Backups**. If you enable automated backups, follow the instructions in the Backups topic to configure the backups.

2. Ensure that the **Instances** setting for **Backup Prepare Node** in the **Resource Config** pane matches your automated backups enable or disable choice:
    - **Disable Backups** selected: set **Backup Prepare Node** > **Instances** to `0`.
    - **Enable Backups** selected: set **Backup Prepare Node** > **Instances** to `1`.

Advanced Options



The **Advanced Options** pane lets you configure the following features:

- The Replication Canary: For more information, see Monitoring the MySQL Service.
- The Interruptor: For more information, see Using the Interruptor.
- **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. Quota Enforcer polling draws minimal resources, but if you want to reduce this load, increase this interval. Be aware, however, that a long Quota Enforcer interval may cause apps to write more data than their pre-determined limit allows.
- **Cluster Name**
  Default: `cf-mariadb-galera-cluster`
  Set a unique name for the cluster. When examining the state or making manual changes to a Pivotal MySQL cluster, it's useful to check the name of the cluster. You can see the name of the cluster by running the query, `select @@wsrep_cluster_name`.

  Important: may only be set during initial deployment. Changing this field after the cluster has been deployed causes **Apply Changes** to fail.

  Please contact Pivotal Support for help in changing the name of a cluster that has already been deployed.

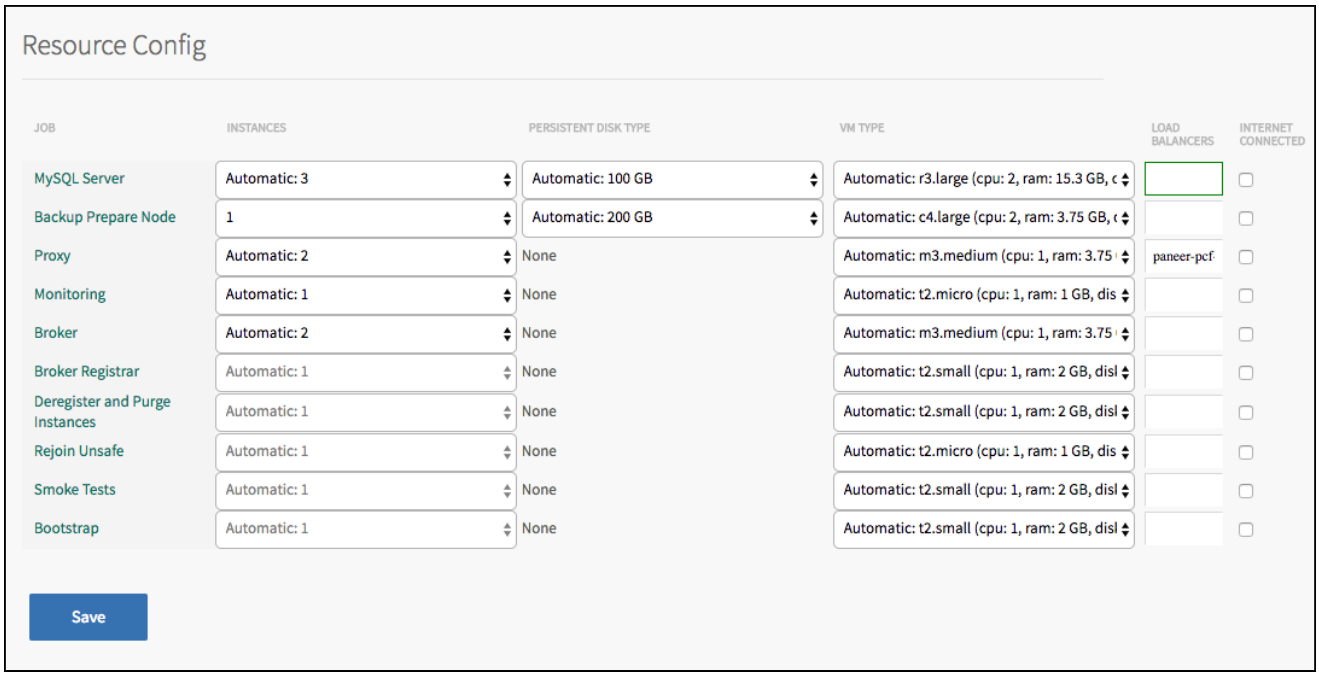
Errands

Two post-deploy errands 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 apps pushed to Elastic Runtime can be bound and write to MySQL service instances. You can turn both errands on or off in the **Errands** pane under the **Settings** tab.

Note: Disabling errands can result in unexpected side effects. Do not reconfigure errands in your deployment without instruction from Pivotal Support. Additionally, the **Errands** pane also shows a **broker-deregistrar** pre-delete errand. Ops Manager runs the broker-deregistrar errand to

clean up when it uninstalls a tile. Running `bosh run errand broker-registrar` under any other circumstances deletes user data. Do not run this errand unless instructed to do so by Pivotal Support.

Resource Config



This pane configures the number, persistent disk capacity, and VM type for all component VMs that run the MySQL for PCF service.

Make sure to provision ample resources for your **MySQL Server** nodes. MariaDB servers require sufficient CPU, RAM, and IOPS to promptly respond to client requests. Also note that the MySQL for PCF reserves about 2-3 GB of each instance's persistent disk for service operations use. The rest of the capacity is available for the databases. The MariaDB cluster nodes are configured by default with 100GB of persistent disk. The deployment fails if this is less than 3GB; we recommend allocating 10GB minimum.

If **Enable Backups** is selected in the **Backups** pane, set **Backup Prepare Node** > **Instances** to `1`.

Switch Between Single and HA Topologies

To switch your MySQL for PCF service between single-node and high availability (HA) topologies, navigate to the **Resource Config** pane and change the **Instances** settings for the **MySQL Server**, **Proxy**, and **Service Broker** components as shown in this table:

Resource	Single-Node	High Availability (HA)
MySQL Server	1	3
Proxy	1	2
Service Broker	1	1-2*

*Routine database operations do not require two service brokers.

Single and three-node clusters are the only supported topologies. Ops Manager allows you to set the number of **MySQL Server** instances to other values, but Pivotal recommends only one or three.

If you scale up to three MySQL nodes, Pivotal recommends spreading them across different Availability Zones to maximize cluster availability. An Availability Zone (AZ) is a network-distinct section of a region. For more information about Amazon AZs, see Amazon's documentation.

When you change the instance counts for a MySQL service, a top-level property is updated with the new nodes' IP addresses. As BOSH deploys, it updates 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 causes all connections to that node to be dropped while the node restarts.

Stemcell

This pane uploads the stemcell that you want the service components to run on. Find available stemcells at Pivotal Network.

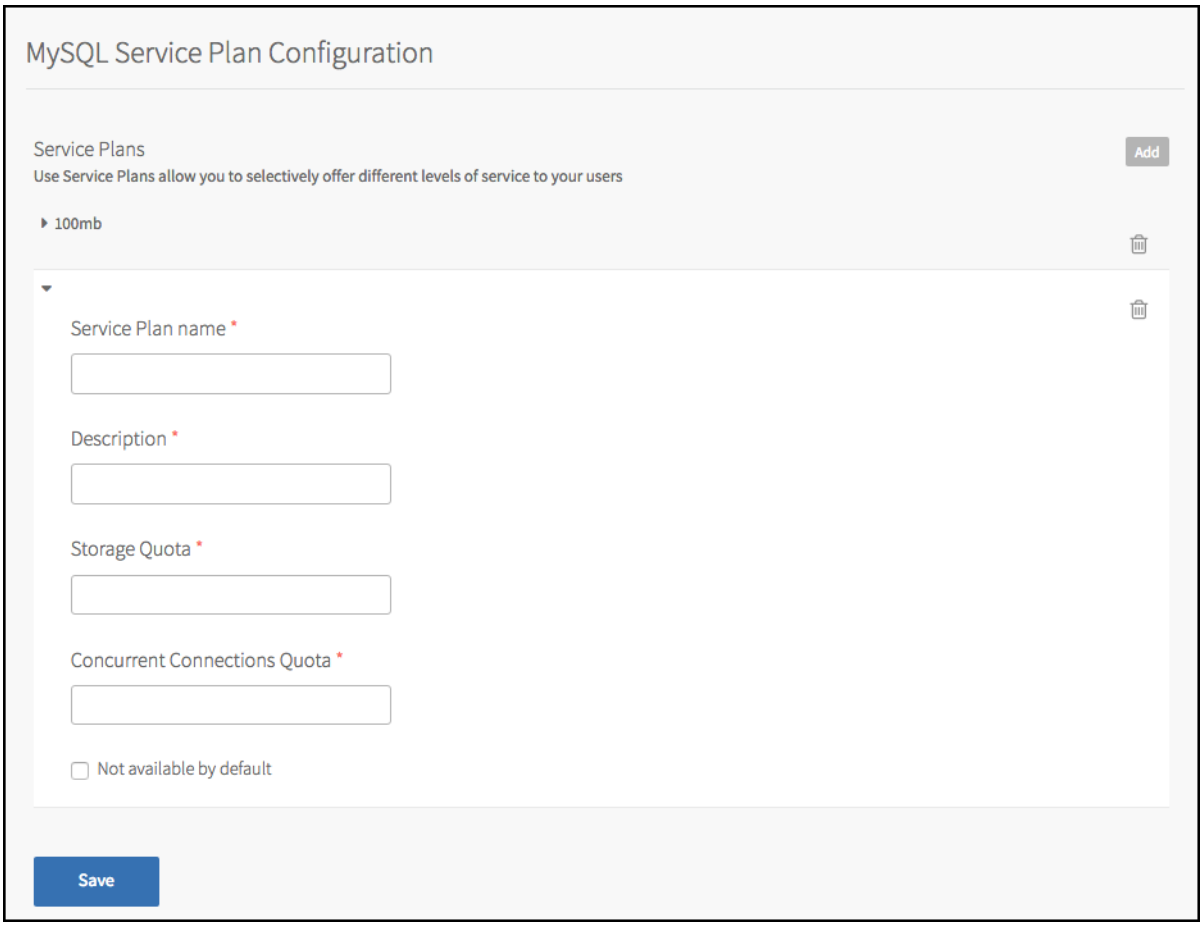
Service Plans

Operators can configure multiple MySQL service plans in the **Service Plans** pane of the Ops Manager MySQL tile. Follow the instructions below to add, modify, or delete service plans.

Note: Prior to v1.8.0, MySQL for PCF supported only one service plan. Upgrading from v1.7.x or earlier renames this plan to `pre-existing-plan` by default. To retain the original name of your plan (e.g., “100mb-dev”), edit its **Service Plan name** field in the **Service Plans** pane before clicking **Apply Changes**

Add a Plan

1. Navigate to the Ops Manager Installation Dashboard and click the **MySQL for PCF** tile.
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.



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

      Note: Ops Manager does not prevent you from entering invalid names into the **Service Plan name** field. A plan name containing invalid characters prevents the tile from deploying after you click **Apply Changes** and generates an error like this in the installation log: `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**: This descriptive text accompanies the plan name to provide context. 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.
    - **Not available by default**: By default, plans are *public* and publish to all orgs. Selecting this checkbox makes the plan *private*, which means the

operator must publish the plan manually before developers can use it. See Access Control for how to change the scope of publication for service plans already deployed.

- The **Concurrent Connections Quota** field sets the `max_user_connections` property for an existing plan.

Changing the **Concurrent Connections Quota** does not affect the connections currently open. For example, if you decrease this value from 40 to 20, apps already running with 40 open connections will retain their connections. To force an app's open connection count down to the new limit, an operator can restart the proxy job. Otherwise, the number of connections will eventually converge down to the new limit on its own, because when any app connection above the limit is reset, it won't reconnect.

Note: You cannot deploy MySQL for PCF without at least one plan defined. If you want to deploy the MySQL tile so that no plans are visible to developers, define one plan, select **Not available by default** to make the plan private, and only enable access to your own org.

Modify a Plan

To modify an existing service plan, change its configuration values in the Ops Manager **MySQL** tile **Service Plans** pane, click **Save** and then click **Apply Changes** in the Installation Dashboard.

Do not decrease the **Storage Quota** value for a plan, and only decrease the **Concurrent Connections Quota** value if you are confident that all apps using the plan use fewer concurrent connections than allowed. Reducing these values could cause app failure when existing service instances update. Instead of decreasing quotas for a plan, create a new plan with lower quotas.

After an operator changes a plan's definition, either the operator or a user must update each plan instance by running `cf update-service SERVICE_INSTANCE -p NEW_PLAN_NAME` on the command line.

Update Existing Service Instances

After you change a service plan, all new services reflect the new settings. To apply a change to existing services, update service instances using the cf CLI as follows:

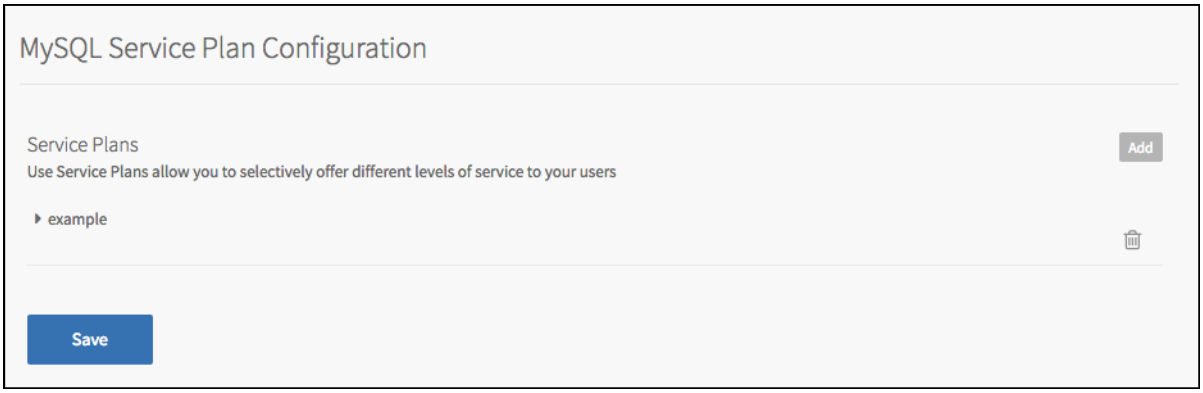
```
cf update-service SERVICE_INSTANCE -p NEW_PLAN
```

The following rules apply when updating a service instance's plan:

- You can always update a service instance to a plan with a larger `max_storage_mb`.
- You can update a service instance to a plan with a smaller `max_storage_mb` only if the current usage is less than the new value. If current usage is greater than the new value, the `update-service` command fails.

Delete a Plan

1. Navigate to the Ops Manager Installation Dashboard and click the **MySQL for PCF** tile.
2. Click **Service Plans**.



3. Click the trash can icon to the right of the service plan listing you want to delete.

> **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.

The plan will no longer appear in the Services Marketplace for non-admin users. Existing service instances will continue to exist until deleted.

> **Note**: If you want to recover a service plan that you deleted accidentally, see Accidental Deletion of Service Plan.

Instances Left Over After Plan Deletion

When deleting a plan, the **Apply Changes** step will run the `broker-registrar` errand. If there are still services instances that were created under that plan, the errand will show this warning:

```
Warning: Service plans are missing from the broker's catalog (BROKER_CATALOG_URL)
but can not be removed from Cloud Foundry while instances exist. The plans have
been deactivated to prevent users from attempting to provision new instances of
these plans. The broker should continue to support bind, unbind, and delete for
existing instances; if these operations fail contact your broker provider.
```

You can see how many instances of each plan are allocated by running this SQL as the `admin` user:

```
SELECT max_storage_mb AS plan_size,
    COUNT(db_name) AS total_instances_allocated
FROM mysql_broker.service_instances
GROUP BY max_storage_mb
ORDER BY max_storage_mb DESC;
```

After all service plan instances have been deleted, remove the plan from the Marketplace by running the `broker-registrar` errand again:

1. Find the full name of the MySQL for PCF deployment. For example, `p-mysql-180290d67d5441ebf3c5`.

   ```
   bosh deployments
   ```

2. Set the deployment:

   ```
   bosh deployment p-mysql-180290d67d5441ebf3c5
   ```

3. Run:

   ```
   bosh run errand broker-registrar
   ```

Re-adding Deleted Plans

A plan name can only be reused if the old plan has been fully purged using the instructions above.

If there are still service instances for a plan that has been deleted, the `broker-registrar` errand fails with the following error:

```
Server error, status code: 502, error code: 270012, message: Service broker
catalog is invalid: Plan names must be unique within a service. Service p-mysql
already has a plan named
```

MySQL Proxy

MySQL for Pivotal Cloud Foundry (PCF) uses the Switchboard router to proxy TCP connections to healthy MariaDB nodes.

Using a proxy gracefully handles failure of MariaDB nodes, enabling fast, unambiguous failover to other server nodes within the cluster. When a server node becomes unhealthy, the proxy closes all connections to the unhealthy node and re-routes all subsequent connections to a healthy server node.

On-demand versions of MySQL for PCF do not use proxies.

Switchboard offers three separate functions, for which it uses three separate ports. The default ports are documented, along with all other network port requirements, on the installation page.

- **MySQL Server Access**
  MySQL clients communicate with the MySQL servers through this network port. Do not configure any load balancer to check Proxy health via this port; these connections will be automatically passed through to the MySQL servers.
- **Proxy Health**
  Switchboard listens for connections on Port 1936. When using a load balancer to balance across multiple instances of Switchboard, configure the load balancer to test for health by connecting to port 1936. To support different kinds of load balancers, Switchboard supports simple TCP connection checks and HTTP requests. HTTP connections are not authenticated, and require no special load balancer configuration.
- **Proxy Dashboard and API**
  Operators can connect to Switchboard to view the state of the cluster back-ends. Read either the Dashboard or API documentation for more information.

Node Health

When determining where to route traffic, the proxy queries an HTTP healthcheck process running on the database node VM. This healthcheck can return as either healthy or unhealthy, or the node can be unresponsive.

Healthy

If the healthcheck process returns HTTP status code `200`, the proxy includes the node in its pool of healthy nodes.

When a new or resurrected nodes rejoin the cluster, the proxy continues to route all connections to the currently active node. In the case of failover, the proxy considers all healthy nodes as candidates for new connections.

Unhealthy

If the healthcheck returns HTTP status code `503`, the proxy considers the node unhealthy.

This happens when a node becomes non-primary, as specified by the cluster behavior section of the *Architecture* topic.

The proxy severs existing connections to newly unhealthy node. The proxy routes new connections to a healthy node, assuming such a node exists. Clients are expected to handle reconnecting on connection failure should the entire cluster become inaccessible.

Unresponsive

If node health cannot be determined due to an unreachable or unresponsive healthcheck endpoint, the proxy considers the node unhealthy. This may happen if there is a network partition or if the VM running the MariaDB node and healthcheck died.

Proxy Count

If the operator sets the total number of proxy hosts to `0` in OpsManager or BOSH deployment manifest, apps connect directly to one MariaDB server node. This makes that node a single point of failure (SPOF) for the cluster.

For high-availability, Pivotal recommends running two proxies, which provides redundancy should one of the proxies fail.

Proxy Dashboard

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

From a browser, you can open the dashboard for each proxy instance at: `http://JOB-INDEX-proxy-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`, you would access your proxy instance dashboards would at:

- https://0-proxy-p-mysql.example.com
- https://1-proxy-p-mysql.example.com

> **Note**: Earlier versions of MySQL for PCF use a different hostname format for the proxies, in the form: http://proxy-JOB-INDEX-p-mysql.SYSTEM-DOMAIN
> For example: https://proxy-0-p-mysql.example.com

You need basic auth credentials to access these dashboards. You can find them in the **Credentials** tab of the **MySQL for PCF** tile in Ops Manager.

API

The proxy hosts a JSON API at `JOB-INDEX-proxy-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
  }
]
```

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

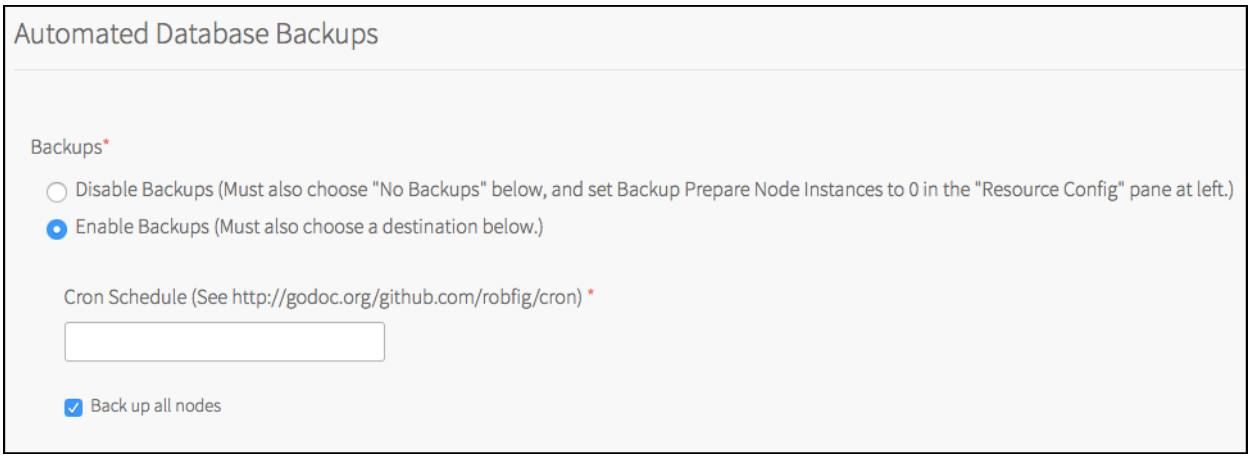
Enable Automated Backups

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

- **How and Where**: There are four 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.
  - MySQL for PCF runs an Azure client that saves backups to a Azure blob store container.
  - MySQL for PCF runs a GCS client that saves backups to a Google Cloud Storage bucket.
- **When**: Backups follow a schedule that you specify with a cron 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 the tile to open the configuration settings.
3. Under the **Settings** tab, click **Backups**. The **Automated Database Backups** pane opens.
4. Under **Backups**, click **Enable Backups**.



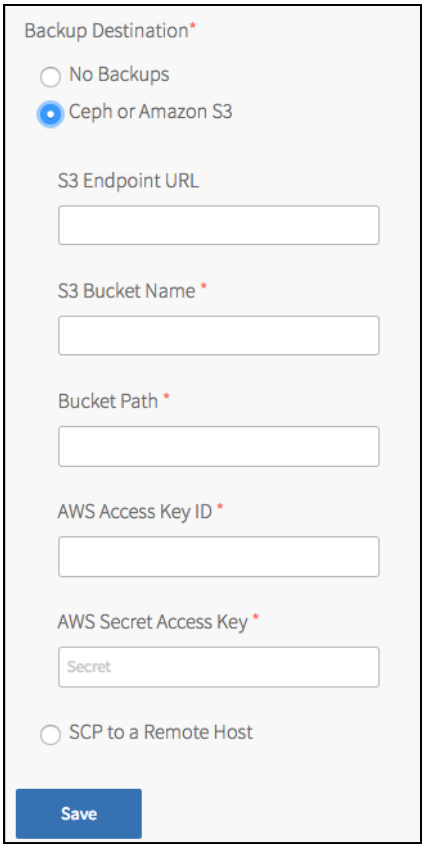
5. For **Cron Schedule**, enter a cron schedule for the backups. The syntax is similar to traditional cron, with additional features such as `@every 1d`, which specifies daily backups. See the cron Go library documentation for more information.
6. If you want to back up all nodes, select the **Back up all nodes** checkbox.

7. Configure a destination where backup artifacts will be delivered. Skip to the appropriate section below:
    - Ceph or AWS
    - Azure
    - Google Cloud Storage
    - SCP
8. Under the **Settings** tab, click **Resource Config**.
9. Set the number of instances for **Backup Prepare Node** to `1`.
10. Click **Save**.

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**.



2. Enter your **S3 Endpoint URL**, for example, `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.

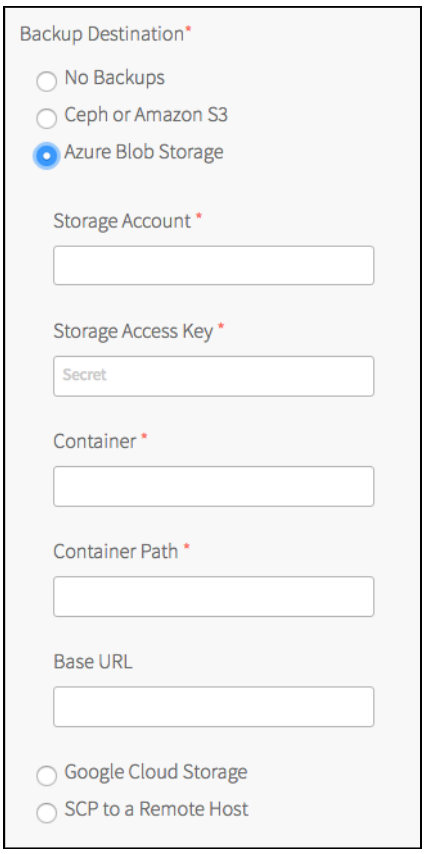
    Note: You must use this folder exclusively for this cluster's backup artifacts. Mixing the backup artifacts from different clusters within a single folder can cause confusion and possible inadvertent loss of backup artifacts.

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**.

Azure

To back up your database on Azure, do the following:

1. Select **Azure Blob Storage**.



2. Enter your **Azure Storage Account**, for example, `mystorageaccount`.
3. Enter your **Azure Storage Access Key**.
4. Enter your **Container** name, for example, `mycontainer`.
5. For **Container Path**, specify a folder within the container 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.

6. (Optional) Configure **Base URL** to specify the base URL for your on-premise blobstore.

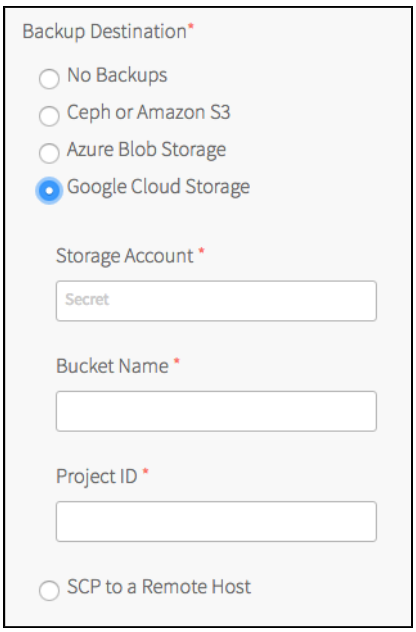
   If you don't specify a URL here, by default, backups are sent to the public Azure blobstore.

7. Click **Save**.

Google Cloud Storage

To back up your database on Google Cloud Storage (GCS), perform the following steps:

1. Select **Google Cloud Storage**.



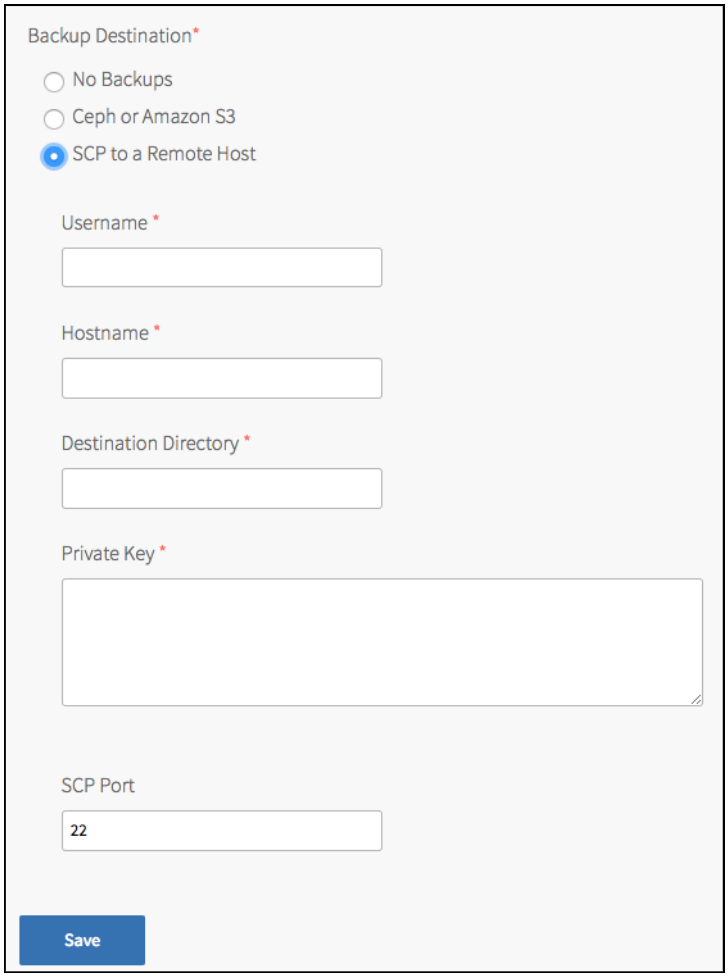
2. Enter your **Google Cloud Storage Account**, for example, `mystorageaccount@google.com`.

> **Note**: Service accounts must have Storage Admin IAM permissions. See documentation for how to generate the JSON key.

3. Enter your **Google Cloud Storage Bucket Name**, for example, `mybucket`.
4. Enter your **Google Cloud Storage Project ID**, for example, `my-project-id`.
5. Click **Save**.

SCP

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



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

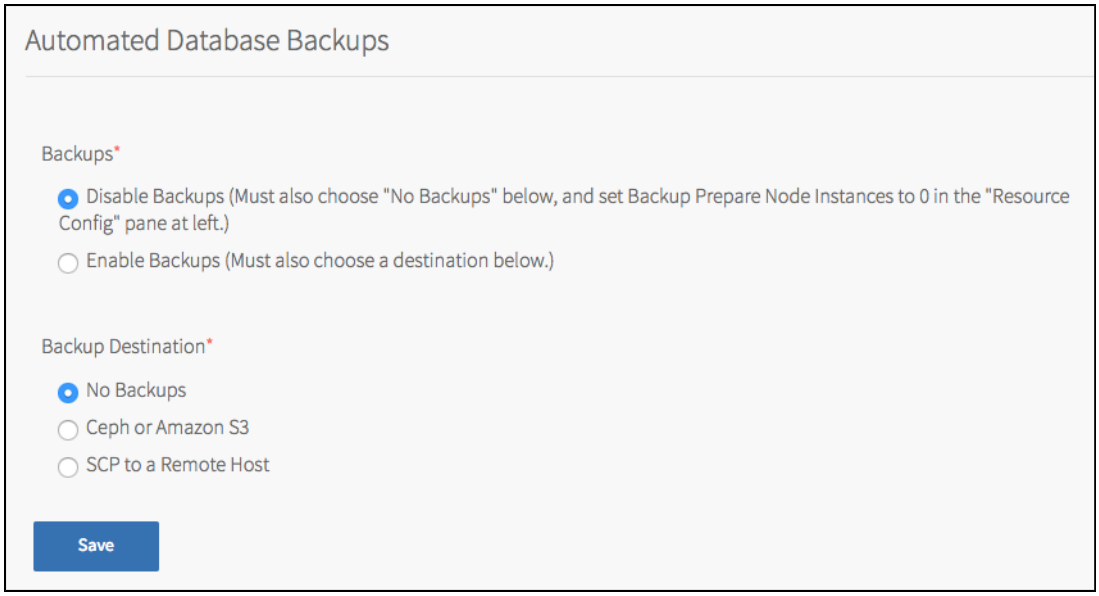
    Note: Pivotal recommends using a VM not within the PCF deployment for the destination of SCP backups. SCP enables the operator to use any desired storage solution on the destination VM.

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.



2. Click **Backups**.
3. Under **Backups**, click **Disable Backups**.
4. Under **Backup Destination**, click **No Backups**.
5. Click **Save**.
6. In the left navigation, click **Resource Config**.
7. Change the number of instances for **Backup Prepare Node** from `1` to `0`.
8. Click **Save**.
9. Return to the Ops Manager Installation Dashboard and click **Apply Changes**.

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

1. Navigate to the MySQL for Pivotal Cloud Foundry tile on the Ops Manager Installation Dashboard.
2. 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`	Host VM
Service Backup	`service-backup`	

Streaming Backup client	`streaming-backup-client`	Backup Prepare VM
Streaming Backup tool	`streaming-backup-tool`	MySQL VM
MySQL server	`mysql`	

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.

Note: To restore a single service instance, see the Restoring a Single Service Instance topic.

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:

    a. 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.
    b. Retrieve the IP address for the MySQL server by navigating to the **MySQL for PCF** tile and clicking the **Status** tab.
    c. Retrieve the VM credentials for the MySQL server by navigating to the **MySQL for PCF** tile and clicking the **Credentials** tab.
    d. 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.
    e. On the MySQL server VM, become super user:

```
$ sudo su
```

    f. Pause the local database server:

```
$ monit stop all
```

    g. Confirm that all jobs are listed as `not monitored`:

```
$ watch monit summary
```

    h. 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:

    a. Move the compressed backup file to the node using `scp`.
    b. 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 -
```

    c. 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
```

    d. Start all services with `monit`:

```
$ monit start all
```

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

```
$ watch monit summary
```

    f. Exit out of the MySQL node.

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

    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 `3`.
    d. Click **Save**.
    e. Return to the Ops Manager Installation Dashboard and click **Apply Changes**.

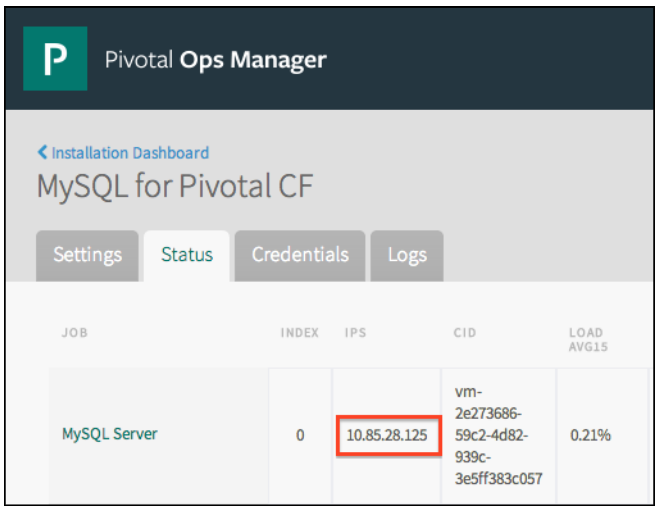
Perform Manual Backup

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:

1. From the Ops Manager Installation Dashboard, click the **MySQL for PCF** tile.
2. Click the **Status** tab.
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

The procedure for backing up your data manually varies depending on whether you have enabled remote admin access to MySQL databases.

In MySQL for PCF v1.9 and v1.10, remote admin access is disabled by default. However, an operator can enable remote admin access when configuring the MySQL for PCF tile.

To determine whether remote admin access is enabled, perform the following steps:

1. From the Ops Manager Installation Dashboard, click the MySQL for PCF tile.
2. Click **MySQL Server Configuration**.
3. Locate the **Allow remote admin access** checkbox and determine whether or not it is enabled.
   For more information, see MySQL Server Configuration.
4. Use the following table to determine which procedure to follow for your manual backup:

If remote admin access is…	Then follow…
disabled	Remote Admin Access Disabled (Default)
enabled	Remote Admin Access Enabled

Manual Backup with Remote Admin Access Disabled (Default)

Perform the following steps to back up your MySQL for PCF data manually with remote admin access disabled:

1. Use the Cloud Foundry Command Line Interface (cf CLI) to target the Cloud Controller of your PCF deployment with `cf api api.YOUR-SYSTEM-DOMAIN`. For example:

   ```
   $ cf api api.sys.cf-example.com
   ```

   For more information about installing and using the cf CLI, see the cf CLI documentation.

2. Log in:

   ```
   $ cf login
   ```

3. Create a service key for the MySQL service instance. Run the following command: `cf create-service-key SERVICE-INSTANCE-NAME SERVICE-KEY-NAME`

Where:

- `SERVICE-INSTANCE-NAME` : Enter the name of the existing MySQL service instance that contains the data you want to back up.
- `SERVICE-KEY-NAME` : Choose a name for the new service key.

For example:

```
$ cf create-service-key mysql-spring spring-key
Creating service key spring-key for service instance mysql-spring as admin...
OK
```

4. After creating the service key, retrieve its information. Run the following command: `cf service-key SERVICE-INSTANCE-NAME SERVICE-KEY-NAME`

   Where:

   - `SERVICE-INSTANCE-NAME` : Enter the name of the MySQL service instance you created a service key for.
   - `SERVICE-KEY-NAME` : Enter the name of the newly created service key.

   For example:

```
$ cf service-key mysql-spring spring-key
Getting key spring-key for service instance mysql-spring as admin...


{
 "hostname": "10.10.10.5",
 "jdbcUrl": "jdbc:mysql://10.10.10.5:3306/cf_e2d148a8_1baa_4961_b314_2431f57037e5?user=abcdefghijklm\u0026password=123456789",
 "name": "cf_e2d148a8_1baa_4961_b314_2431f57037e5",
 "password": "123456789",
 "port": 3306,
 "uri": "mysql://abcdefghijklm:123456789@10.10.10.5:3306/cf_e2d148a8_1baa_4961_b314_2431f57037e5?reconnect=true",
 "username": "abcdefghijklm"
}
```

5. Examine the output and record the following values:
   - `hostname` : The MySQL for PCF proxy IP address
   - `password` : The password for the user that can be used to perform backups of the service instance database
   - `username` : The username for the user that can be used to perform backups of the service instance database

6. Use mysqldump to back up the data for your service instance. Run the following command: `mysqldump -u USERNAME -p PASSWORD -h MYSQL-PROXY-IP --all-databases --single-transaction > BACKUP-NAME.sql`

   Where:

   - `USERNAME` : Enter the username retrieved from the output of `cf service-key` .
   - `PASSWORD` : Enter the password retrieved from the output of `cf service-key` .
   - `MYSQL-PROXY-IP` : Enter the value of `hostname` retrieved from the output of `cf service-key` .
   - `BACKUP-NAME` : Enter a name for the backup file.

   For example:

```
$ mysqldump -u abcdefghijklm \
-p 123456789 \
-h 10.10.10.8 \
--all-databases \
--single-transaction > spring-db-backup.sql
```

Manual Backup with Remote Admin Access Enabled

If remote admin access is enabled, use mysqldump to back up the data as the admin user.

Note: 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, run the following command:

```
mysqldump -u admin -p PASSWORD -h MYSQL-NODE-IP --all-databases --single-transaction > BACKUP-NAME.sql
```

Where:

- `PASSWORD` : Enter the admin password you retrieved in the Retrieve IP Address and Credentials section.

- `MYSQL-NODE-IP` : Enter the MySQL node IP address you retrieved in the Retrieve IP Address and Credentials section.
- `BACKUP-NAME` : Enter a name for the backup file.

For example:

```
$ mysqldump -u admin -p 123456789 \
  -h 10.10.10.8 --all-databases \
  --single-transaction > user_databases.sql
```

To back up a single database, run the following command: `mysqldump -u admin -p PASSWORD -h MYSQL-NODE-IP DB-NAME --single-transaction > BACKUP-NAME.sql`

Where:

- `PASSWORD` : Enter the admin password you retrieved in the Retrieve IP Address and Credentials section.
- `MYSQL-NODE-IP` : Enter the MySQL node IP address you retrieved in the Retrieve IP Address and Credentials section.
- `DB-NAME` : Enter the name of the database you want to back up.
- `BACKUP-NAME` : Enter a name for the backup file.

For example:

```
$ mysqldump -u admin -p 123456789 \
-h MYSQL-NODE-IP DB-NAME \
--single-transaction > user_databases.sql
```

Manual Restore

Note: This section describes how to restore multiple MySQL service instances. To restore a single service instance, see the Restoring a Single Service Instance topic.

The procedure for restoring your data from a manual backup varies depending on whether you have enabled remote admin access to MySQL databases.

In MySQL for PCF v1.9 and v1.10, remote admin access is disabled by default. However, an operator can enable remote admin access when configuring the MySQL for PCF tile.

To determine whether remote admin access is enabled, perform the following steps:

1. From the Ops Manager Installation Dashboard, click the MySQL for PCF tile.
2. Click **MySQL Server Configuration**.
3. Locate the **Allow remote admin access** checkbox and determine whether or not it is enabled.

   For more information, see MySQL Server Configuration.

Keep in mind the following:

- If you want to restore an entire database, then you must log in as admin.
- The command for restoring your MySQL data from a manual backup is the same for restoring a backup of one or multiple databases. 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.

Restore MySQL Data From a Manual Backup

Perform the following steps to restore your MySQL data from a manual backup:

1. If running in HA configuration, reduce the size of the MySQL for PCF cluster to a single node, following the procedures in theRestore a Backup

Artifact section above.

2. If remote admin access is disabled, perform steps 1 through 5 of the Manual Backup with Remote Admin Access Disabled (Default) section above.

3. Use the MySQL client to enable the creation of tables using any storage engine. Run the following command: `mysql -u USERNAME -p PASSWORD -h MYSQL-PROXY-IP -e "SET GLOBAL enforce_storage_engine=NULL"`

   Where:
   - `USERNAME`:
     - If remote admin access is disabled, enter the username you retrieved from the output of `cf service-key` in the Manual Backup with Remote Admin Access Disabled (Default) section.
     - If remote admin access is enabled, enter the admin password you retrieved in the Retrieve IP Address and Credentials section.
   - `PASSWORD`:
     - If remote admin access is disabled, enter the password you retrieved from the output of `cf service-key` in the Manual Backup with Remote Admin Access Disabled (Default) section.
     - If remote admin access is enabled, enter the admin password you retrieved in the Retrieve IP Address and Credentials section.
   - `MYSQL-PROXY-IP`:
     - If remote admin access is disabled, enter the value of `hostname` you retrieved from the output of `cf service-key` in the Manual Backup with Remote Admin Access Disabled (Default) section.
     - If remote admin access is enabled, enter the MySQL node IP address you retrieved in the Retrieve IP Address and Credentials section.

   For example:

   ```
   $ mysql -u abcdefghijklm \
   -p 123456789 \
   -h 10.10.10.8 -e "SET GLOBAL enforce_storage_engine=NULL"
   ```

4. Use the MySQL client to restore the MySQL database or databases. Run the following command: `mysql -u USERNAME -p PASSWORD -h MYSQL-PROXY-IP < BACKUP-NAME.sql`

   Where:
   - `USERNAME`: Enter the same value as above.
   - `PASSWORD`: Enter the same value as above.
   - `MYSQL-PROXY-IP`: Enter the same value as above.
   - `BACKUP-NAME`: Enter the file name of the backup artifact.

   For example:

   ```
   $ mysql -u abcdefghijklm \
   -p 123456789 \
   -h 10.10.10.8 -e < user_databases.sql
   ```

5. Use the MySQL client to restore the original storage engine restriction. Run the following command: `mysql -u USERNAME -p PASSWORD -h MYSQL-PROXY-IP -e "SET GLOBAL enforce_storage_engine='InnoDB'"`

   Where:
   - `USERNAME`: Enter the same value as above.
   - `PASSWORD`: Enter the same value as above.
   - `MYSQL-PROXY-IP`: Enter the same value as above.

   For example:

   ```
   $ mysql -u abcdefghijklm \
   -p 123456789 \
   -h 10.10.10.8 -e "SET GLOBAL enforce_storage_engine='InnoDB'"
   ```

6. If you are running in HA mode, re-configure MySQL for PCF to run using three nodes by following the procedures in the Restore a Backup Artifact section above.

   If you are not running HA mode, restart the database server. This step is not necessary if scaling back to three MySQL nodes. Run the following commands:

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

For more examples of manual backup and restore procedures, see the MariaDB documentation.

Restoring a Single Service Instance

This topic describes how to restore a single service instance of MySQL for Pivotal Cloud Foundry (PCF).

Overview

Because automated backups for MySQL for PCF back up all data stored by the cluster, each backup artifact contains all of the service instances. For more information about automated backups, see the Enable Automated Backups section of the *Backups* topic.

Backups of the entire cluster are useful for operators dealing with a disaster recovery scenario. But in cases where user error invalidates a single service instance, the operator might want to restore the data for only one instance.

Restore Your Service Instance

Perform the following procedures to restore a single service instance:

1. Deploy a MariaDB server in a different environment than the one that contains your MySQL for PCF cluster. Pivotal recommends deploying the MySQL for PCF tile to a non-production PCF environment or deploying the official Docker image of MariaDB to a Docker container. For more information about deploying MySQL for PCF, see Installing MySQL for PCF. For more information about deploying MariaDB with Docker, see the MariaDB documentation.

2. Locate your backup artifact. The location of your backup artifact depends on how you configured your automated backups. For more information, see the Enable Automated Backups section of the *Backups* topic.

3. Transfer the backup artifact to the temporary MariaDB instance.

4. SSH into the temporary MariaDB instance.

5. Delete the existing MySQL data that is stored on the temporary MariaDB instance by running `rm -rf PATH-TO-MARIADB-DATA-DIRECTORY/*`. For example,

   ```
   $ rm -rf /mysql/*
   ```

   If you are using MySQL for PCF, the path to your MariaDB data directory is `/var/vcap/store/mysql`.

6. Decrypt and expand the backup artifact into the MariaDB data directory. For example,

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

7. Change the owner of the MariaDB data directory, because MySQL expects the data directory to be owned by a particular user. For example,

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

8. Restart MariaDB using the method you used to deploy the temporary MariaDB server.

9. To back up the database that contains the service instance, use `mysqldump` and specify the database name. For example,

   ```
   $ mysqldump -u admin -p -h localhost YOUR-DATABASE-NAME --single-transaction > YOUR-DATABASE-NAME.sql
   ```

10. Perform steps 1 through 5 of Manual Backup with Remote Admin Access Disabled (Default) in *Backups*.

11. Use the MySQL client to restore the MySQL database or databases. Run the following command from your local machine:
    `mysql -u USERNAME -p PASSWORD -h MYSQL-PROXY-IP < BACKUP-NAME.sql`

    Where:
    - `USERNAME`: Enter the username retrieved from the output of `cf service-key`.
    - `PASSWORD`: Enter the password retrieved from the output of `cf service-key`.
    - `MYSQL-PROXY-IP`. Enter the value of `hostname` retrieved from the output of `cf service-key`.
    - `BACKUP-NAME`: Enter the file name of the backup artifact created above with `mysqldump`.

Note: During the restore process, all apps bound to the service instance should be down, or in read-only mode. This is because the data will be inconsistent during this process.

For example:

```
$ mysql -u abcdefghijklm \
-p 123456789 \
-h 10.10.10.8 -e < user_database.sql
```

This replaces the existing data in the MySQL for PCF cluster with the content of the backup artifact, without affecting the data of any other service instances.

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 enable applications to access the service 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 called `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
    ```

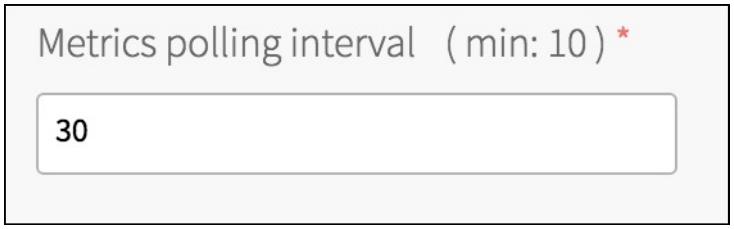
Monitoring the MySQL Service

This document describes the metrics that are produced by MySQL for PCF, and everything you need to know about the Replication Canary. For information about the Interruptor, see Using the Interruptor.

Metrics

MySQL emits a number of metrics that can be used to monitor the health and performance of the MySQL deployment. The Loggregator Firehose exposes these metrics.

The metrics polling interval defaults to 30 seconds. This can be changed by navigating to the Advanced Options configuration pane and entering a new value in **Metrics polling interval (min: 10)**.



Third-party monitoring tools can consume MySQL for PCF's metrics via a nozzle to monitor MySQL performance and health. For an example Datadog configuration that displays some of the significant metrics outlined below, see the CF Redis and MySQL example dashboards. Pivotal does not endorse or provide support for any third party solution.

Key Performance Indicators

Key Performance Indicators (KPIs) for MySQL for PCF are metrics that operators find most useful for monitoring their MySQL service to ensure smooth operation. KPIs are high-signal-value metrics that can indicate emerging issues. KPIs can be raw component metrics or *derived* metrics generated by applying formulas to raw metrics.

Pivotal provides the following KPIs as general alerting and response guidance for typical MySQL for PCF installations. Pivotal recommends that operators continue to fine-tune the alert measures to their installation by observing historical trends. Pivotal also recommends that operators expand beyond this guidance and create new, installation-specific monitoring metrics, thresholds, and alerts based on learning from their own installations.

MySQL for PCF KPIs

This section lists the KPIs that are specific for MySQL for PCF.

For a list of general KPIs that apply to all instances, and not specifically to MySQL for PCF, see BOSH System Metrics.

For a list of all MySQL for PCF component metrics, see All MySQL Metrics.

MySQL Server Availability

	/p-mysql/available
Description	The MySQL Server is currently responding to requests, which indicates that the server is running. **Use:** This metric is especially useful in single-node mode, where cluster metrics are not relevant. If the server does not emit heartbeats, it is offline. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** boolean **Frequency:** 30 s (default)
Recommended measurement	Average over the last 5 minutes

Recommended alert thresholds	Yellow warning: N/A /p-mysql/available **Red critical:** < 1
Recommended response	Run `mysql-diag` and check the MySQL Server logs for errors.

Galera Cluster Node Readiness

	/p-mysql/galera/wsrep_ready
Description	Shows whether each cluster node can accept queries. Returns only 0 or 1. When this metric is 0, almost all queries to that node fail with the error: `ERROR 1047 (08501) Unknown Command` **Use:** Discover when nodes of a cluster have been unable to communicate and, thus, unable to accept transactions. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** boolean **Frequency:** 30 s (default)
Recommended measurement	Average of values of each cluster node, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 1.0 **Red critical:** 0 (cluster is down)
Recommended response	- Run `mysql-diag` and check the MySQL Server logs for errors. - Make sure there has been no infrastructure event that affects intra-cluster communication. - Ensure that `wsrep_ready` has not been set to off by using the query: `SHOW STATUS LIKE 'wsrep_ready';`

Galera Cluster Size

	/p-mysql/galera/wsrep_cluster_size
Description	The number of cluster nodes with which each node is communicating normally. **Use:** When running in a multi-node configuration, this metric indicates if each member of the cluster is communicating normally with all other nodes. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** count **Frequency:** 30 s (default)
Recommended measurement	(Average of the values of each node / cluster size), over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 3.0 (availability compromised) **Red critical:** < 1.0 (cluster unavailable)
Recommended response	Run `mysql-diag` and check the MySQL Server logs for errors.

Galera Cluster Status

	/p-mysql/galera/wsrep_cluster_status
Description	Shows the primary status of the cluster component that the node is in. Values are: - Primary = 1 - Non-primary = 0 - Disconnected = -1 See: https://mariadb.com/kb/en/mariadb/galera-cluster-status-variables/ **Use:** Any value other than “Primary” indicates that the node is part of a nonoperational component. This occurs in cases of multiple membership changes that result in a loss of quorum.

	/p-mysql/galera/wsrep_cluster_status
	Envelope Type: Gauge **Unit:** integer (see above) **Frequency:** 30 s (default)
Recommended measurement	Sum of each of the nodes, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 3 **Red critical:** < 1
Recommended response	- Check node status to ensure that they are all in working order and able to receive write-sets. - Run `mysql-diag` and check the MySQL Server logs for errors.

Persistent and Ephemeral Disk Free

	/p-mysql/system/persistent_disk_free and /p-mysql/system/ephemeral_disk_free
Description	The number of megabytes that are available on the file systems. **Use:** MySQL cannot function correctly if there isn't sufficient free space on the file systems. Use these metrics to ensure that your drives have disks large enough for your user base. **Origin:** Doppler/Firehose **Type:** Megabytes **Frequency:** 30 s (default)
Recommended measurement	The minimum disk free of all of the nodes / persistent disk size
Recommended alert thresholds	**Yellow warning:** < 80% **Red critical:** < 90%
Recommended response	- Audit plan allocation by your users, recommend deletion of unused service instances. - Redeploy with larger disks.

Service Plans Allocated

	/p-mysql/broker/disk_allocated_service_plans
Description	The number of megabytes allocated by the broker for all service plans, current and allocated. **Use:** The service broker will not allow new service instances to be created if there isn't sufficient unreserved space remaining on the persistent disk. **Origin:** Doppler/Firehose **Type:** Megabytes **Frequency:** 30 s (default)
Recommended measurement	disk allocated / persistent disk size
Recommended alert thresholds	**Yellow warning:** < 80% **Red critical:** < 90%
Recommended response	- Audit plan allocation by your users, recommend deletion of unused service instances. - Redeploy with larger persistent disks

Connections per Second

	/p-mysql/net/connections
Description	Connections per second made to the server. **Use:** If the number of connections drastically changes or if apps are unable to connect, there might be a network or app issue. **Origin:** Firehose **Envelope Type:** Gauge

	/p-mysql/net/connections
	Unit: count **Frequency:** 30 s (default)
Recommended measurement	The maximum number of connections of any node / `max connections`, over last 1 minute
Recommended alert thresholds	**Yellow warning:** > 80% **Red critical:** > 90%
Recommended response	- Run `mysql-diag` and check the MySQL Server logs for errors. - When approaching 100% of max connections, Apps may be experiencing times when they cannot connect to the database. The connections per second for the cluster vary based on application instances and app utilization. If this threshold is met or exceeded for an extended period of time, monitor app usage to ensure everything is behaving as expected.

Query Rate

	/p-mysql/performance/questions
Description	The number of statements executed by the server, excluding statements executed within stored programs. **Use:** The cluster should always be processing some queries, if just as part of the internal automation. **Origin:** Firehose **Envelope Type:** Gauge **Unit:** count **Frequency:** 30 s (default)
Recommended measurement	Change in number between the current and previous polling period
Recommended alert thresholds	**Yellow warning:** 0 for 90 s **Red critical:** 0 for 120 s
Recommended response	If the rate is ever zero for an extended time, run `mysql-diag` and investigate the MySQL server logs to understand why query rate changed and determine appropriate action.

BOSH System Metrics

All BOSH-deployed components generate the following system metrics; these system metrics also serve as KPIs for the MySQL for PCF service.

RAM

	system.mem.percent
Description	RAM being consumed by the MySQL cluster node. **Use:** MySQL increases its memory usage as the data set increases. This is normal, as much of that RAM is used to buffer IO. As long as there is enough remaining RAM for other processes on the instance, the MySQL server should be OK. **Origin:** JMX Bridge or BOSH HM **Type:** percentage **Frequency:** 60 s
Recommended measurement	Average over the last 10 minutes
Recommended alert thresholds	**Yellow warning:** > 95% **Red critical:** > 99%
Recommended response	Update the cluster to use VMs that offer more RAM.

CPU

	system.cpu.percent

	system.cpu.percent
Description	CPU time being consumed by the MySQL cluster. **Use:** A node that experiences context switching or high CPU usage will become unresponsive. This also affects the ability of the node to report metrics. **Origin:** JMX Bridge or BOSH HM **Type:** percent **Frequency:** 60 s
Recommended measurement	Average over the last 10 minutes
Recommended alert thresholds	**Yellow warning:** > 80% **Red critical:** > 90%
Recommended response	Determine what is using so much CPU. If it is from normal processes, update the service instance to use a VM with larger CPU capacity.

Persistent Disk

	persistent.disk.percent
Description	Persistent disk being consumed by the MySQL cluster nodes. **Use:** If the persistent disk fills up, MySQL will be unable to process queries and recovery is difficult. **Origin:** JMX Bridge or BOSH HM **Type:** percent **Frequency:** 60 s
Recommended measurement	Average over the last 10 minutes
Recommended alert thresholds	**Yellow warning:** > 75% **Red critical:** > 90%
Recommended response	Update the deployment with larger persistent disks. This process may take some time, as the data is copied from the original persistent disk to a new one.

MySQL-Specific Metrics

Data Source	Description	Metric Unit
`/p-mysql/available`	Indicates if the local database server is available and responding.	boolean
`/p-mysql/system/persistent_disk_used`	The number of KB used on the persistent disk.	KB
`/p-mysql/system/persistent_disk_free`	The number of KB available on the persistent disk.	KB
`/p-mysql/system/persistent_disk_inodes_used`	The number of inodes used on the persistent disk.	count
`/p-mysql/system/persistent_disk_inodes_free`	The number of inodes available on the persistent disk.	count
`/p-mysql/system/ephemeral_disk_used`	The number of KB used on the ephemeral disk.	KB
`/p-mysql/system/ephemeral_disk_free`	The number of KB available on the ephemeral disk.	KB
`/p-mysql/system/ephemeral_disk_inodes_used`	The number of inodes used on the ephemeral disk.	count
`/p-mysql/system/ephemeral_disk_inodes_free`	The number of inodes available on the ephemeral disk.	count
`/p-mysql/broker/disk_allocated_service_plans`	The number of MB allocated by the broker for all service plans, current and allocated.	MB

Data Source	Description	Metric Unit
/p-mysql/innodb/buffer_pool_free	The number of free pages in the InnoDB Buffer Pool.	pages
/p-mysql/innodb/buffer_pool_total	The total number of pages in the InnoDB Buffer Pool.	pages
/p-mysql/innodb/buffer_pool_used	The number of used pages in the InnoDB Buffer Pool.	pages
/p-mysql/innodb/buffer_pool_utilization	The utilization of the InnoDB Buffer Pool.	fraction
/p-mysql/innodb/current_row_locks	The number of current row locks.	locks
/p-mysql/innodb/data_reads	The rate of data reads.	reads/second
/p-mysql/innodb/data_writes	The rate of data writes.	writes/second
/p-mysql/innodb/mutex_os_waits	The rate of mutex OS waits.	events/second
/p-mysql/innodb/mutex_spin_rounds	The rate of mutex spin rounds.	events/second
/p-mysql/innodb/mutex_spin_waits	The rate of mutex spin waits.	events/second
/p-mysql/innodb/os_log_fsyncs	The rate of fsync writes to the log file.	writes/second
/p-mysql/innodb/row_lock_time	Time spent in acquiring row locks.	milliseconds
/p-mysql/innodb/row_lock_waits	The number of times per second a row lock had to be waited for.	events/second
/p-mysql/net/connections	The rate of connections to the server.	connection/second
/p-mysql/net/max_connections	The maximum number of connections that have been in use simultaneously since the server started.	connections
/p-mysql/performance/com_delete	The rate of delete statements.	queries/second
/p-mysql/performance/com_delete_multi	The rate of delete-multi statements.	queries/second
/p-mysql/performance/com_insert	The rate of insert statements.	query/second
/p-mysql/performance/com_insert_select	The rate of insert-select statements.	queries/second
/p-mysql/performance/com_replace_select	The rate of replace-select statements.	queries/second
/p-mysql/performance/com_select	The rate of select statements.	queries/second
/p-mysql/performance/com_update	The rate of update statements.	queries/second
/p-mysql/performance/com_update_multi	The rate of update-multi.	queries/second
/p-mysql/performance/created_tmp_disk_tables	The rate of internal on-disk temporary tables created by second by the server while executing statements.	table/second
/p-mysql/performance/created_tmp_files	The rate of temporary files created by second.	files/second
/p-mysql/performance/created_tmp_tables	The rate of internal temporary tables created by second by the server while executing statements.	tables/second
/p-mysql/performance/kernel_time	Percentage of CPU time spent in kernel space by MySQL.	percent
/p-mysql/performance/key_cache_utilization	The key cache utilization ratio.	fraction
/p-mysql/performance/open_files	The number of open files.	files
/p-mysql/performance/open_tables	The number of of tables that are open.	tables
/p-mysql/performance/qcache_hits	The rate of query cache hits.	hits/second
/p-mysql/performance/questions	The rate of statements executed by the server.	queries/second
/p-mysql/performance/slow_queries	The rate of slow queries.	queries/second
	The total number of times that a request for a table lock could not be granted	number

Data Source	Description	Metric Unit
/p-mysql/performance/table_locks_waited	immediately and a wait was needed.	
/p-mysql/performance/threads_connected	The number of currently open connections.	connections
/p-mysql/performance/threads_running	The number of threads that are not sleeping.	threads
/p-mysql/performance/max_connections	The maximum permitted number of simultaneous client connections.	integer
/p-mysql/performance/open_files_limit	The number of files that the operating system permits mysqld to open.	integer
/p-mysql/performance/open_tables	The number of tables that are open.	integer
/p-mysql/performance/opened_tables	The number of tables that have been opened.	integer
/p-mysql/performance/opened_table_definitions	The number of .frm files that have been cached.	integer

Galera-Specific Metrics

Data Source	Description	Metric Unit
/p-mysql/galera/wsrep_ready	Shows whether the node can accept queries.	boolean
/p-mysql/galera/wsrep_cluster_size	The current number of nodes in the Galera cluster.	node
/p-mysql/galera/wsrep_cluster_status	Shows the primary status of the cluster component that the node is in.	State ID. Values are Primary = 1, Non-primary = 0, Disconnected = -1 (See: https://mariadb.com/kb/en/mariadb/galera-cluster-status-variables/)
/p-mysql/galera/wsrep_local_recv_queue_avg	Shows the average size of the local received queue since the last status query.	float
/p-mysql/galera/wsrep_local_send_queue_avg	Shows the average size of the local sent queue since the last status query.	float
/p-mysql/galera/wsrep_local_index	This node index in the cluster (base 0).	int
/p-mysql/galera/wsrep_local_state	This is the node's local state	int

For more information on monitoring PCF, see Monitoring Pivotal Cloud Foundry.

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:

- Emails 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 Email

If the Canary detects a replication failure, it immediately sends an email 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 email 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-api-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.

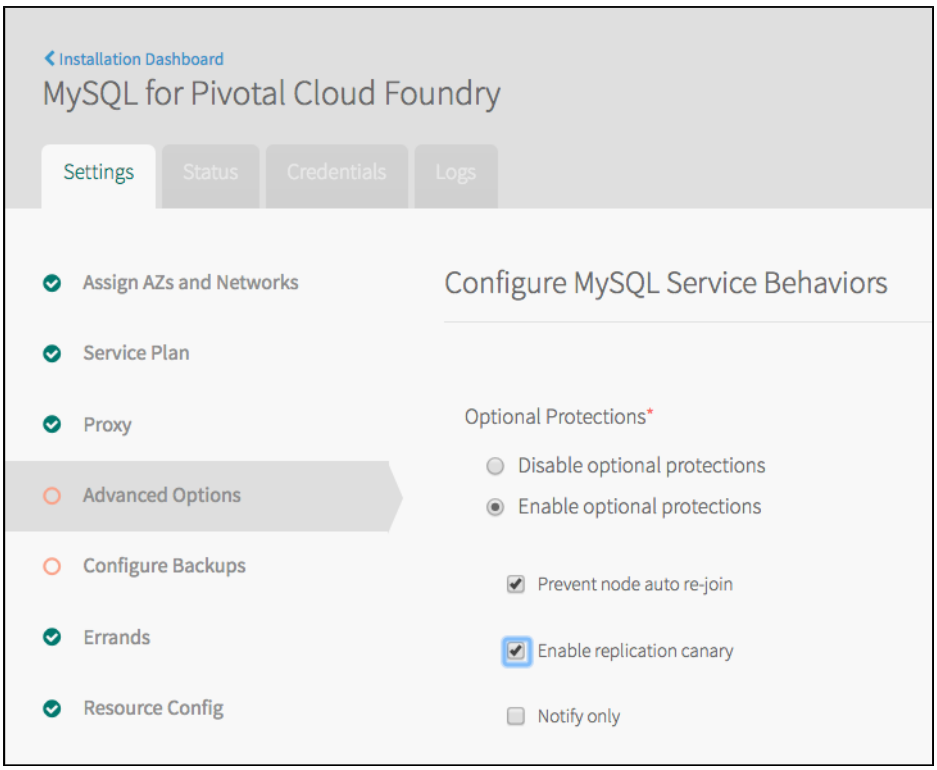
Configure the Elastic Runtime Tile

> **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

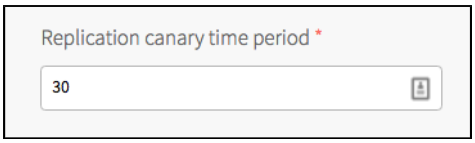
1. In the **Advanced Options** section, select **Enable replication canary**.



2. If you want to the Replication Canary to send email 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.

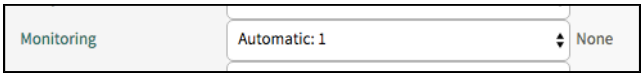
3. You can override the **Replication canary time period**. The **Replication canary time period** sets how frequently the canary checks for replication failure, in seconds. This adds a small amount of load to the databases, but the canary reacts more quickly to replication failure. The default is 30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



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 **Email address** to receive monitoring notifications. Use a closely monitored email 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.



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**.

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

Troubleshooting and Diagnostics

This topic describes how to diagnose and troubleshoot problems with MySQL for Pivotal Cloud Foundry (PCF).

Diagnose Problems

If your cluster is experiencing downtime or is in a degraded state, Pivotal recommends the following workflow for gathering information to diagnose the type of failure the cluster is experiencing.

1. Use the mysql-diag tool to gather a summary of the network, disk, and replication state of each cluster node.
2. Run download-logs against each node in your MySQL cluster, the MySQL proxies, and the MySQL backup-prepare node. It's important to do this before attempting recovery, because any failures in the recovery procedure can result in logs being lost or made inaccessible.
3. Depending on the reported state from `mysql-diag`, the troubleshooting techniques listed below might allow you to recover your cluster.
4. If you are uncertain about the recovery steps to take, submit a ticket through Pivotal Support with the following information:
    - Logs fetched by running the `download-logs` script
    - Output from `mysql-diag`
    - Type of environment the MySQL deployment is running in (Elastic Runtime, MySQL for PCF, or other)
    - Versions of the installed Ops Manager, Elastic Runtime, and MySQL for PCF

For more diagnostic techniques, see Diagnostic Techniques.

Diagnostic Techniques

After performing the procedure in Diagnose Problems, consult the following additional diagnostic techniques to learn more about your cluster.

Check for Lost Quorum

If your cluster has lost quorum, you must bootstrap it.

To determine whether you need to bootstrap your cluster, check whether the cluster has lost quorum. For more information, see Bootstrapping for more information.

Check Cluster State

To check the 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.

To check how many nodes are in the cluster, perform the following command:

```
$ 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.

To restart your cluster, see Bootstrapping.

Check Replication Status

If you see stale data in your cluster, check whether replication is functioning normally.

Perform the following steps to check the replication status:

1. Obtain the IP addresses of your MySQL server by performing the following steps:
    a. From the 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, see SSH into Ops Manager for more information.
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 from the initial step and `YOUR-IDENTITY` with the `identity` value obtained from the second step. When prompted for a password, provide the `password` value obtained from the second step.
    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='I
    ```

    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 dun
    ```

    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 returns the same result, then you can safely proceed to scaling down your cluster to a single node.

Troubleshoot Problems

This section lists symptoms, solutions, and explanations for the following common problems:

- Cluster errors:
  - Many Replication Errors in Logs
  - No Space Left
  - Node Unable to Rejoin
  - Unresponsive Node(s)
- User errors:
  - Accidental Deletion of Service Plan

Many Replication Errors in Logs

Symptom

You see many replication errors in the MySQL logs.

Explanation

Unless the GRA files show a clear execution error, such as running out of disk space, seeing many replication errors in your logs is a normal behavior. The MySQL for PCF team is developing a more advanced monitoring solution to detect the failure case and alert operators in the future.

Occasionally, replication errors in the MySQL logs resemble the following:

```
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_
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 ...'
```

This error occurs when an app issues an `ALTER TABLE` command that fails to apply to the current schema. This is typically the result of user error.

The MySQL node that receives the request processes it normally. If it fails, the node sends the failure to the app and the app must determine the next step. In a Galera cluster, however, all DDL is replicated and all replication failures are logged. Therefore, the bad `ALTER TABLE` command is run by both slave nodes. If it fails, the slave nodes log it as a "replication failure" because they cannot tell the difference.

Cases where a valid DDL works on some nodes yet fails on others are rare. Usually those cases are limited to problems with running out of disk space or working memory.

An article from Percona suggests that the schemata can get out of sync. However, the author had to deliberately switch a node to RSU, which MySQL for PCF never does except during SST. The article offers a demonstration of what is possible, but does not explain how a customer may actually experience this issue in production.

Solution

Disregard the replication errors as a normal behavior. If you see the "ALTER TABLE" error, fix the user error that produced it.

No Space Left

Symptom

You may encounter one or more of the following symptoms when MySQL has run out of space:

- MySQL gives an error that reports it is out of space
- Queries start failing
- Apps are unable to connect
- You are unable to SSH into a node

Explanation

When MySQL runs out of space, it stops functioning normally.

Solution

1. Use the mysql-diag tool to determine how much space you have left. If the disk is too full, the `mysql-diag` won't work.
2. Redeploy with more persistent disk.
3. After redeploying, you may encounter further errors. To troubleshoot these errors, see the other sections in Troubleshoot Problems.

Node Unable to Rejoin

Symptom

A MySQL node is unable to rejoin a cluster.

Explanation

Existing server nodes restarted with `monit` should automatically join the cluster. If a detached existing node fails to join the cluster, it might be because its sequence number (`seqno`) is higher than those of the nodes with quorum.

A higher `seqno` on the detached node indicates that it has recent changes to the data that the primary lacks. Check this by looking at the node's error log at `/var/vcap/sys/log/mysql/mysql.err.log` .

Solution

If the detached node has a higher sequence number than the primary component, perform one of the following procedures to restore the cluster:

- Shut down the healthy, primary nodes and bootstrap from the detached node to preserve its data. See Bootstrapping for more details.
- Abandon the unsynchronized data on the detached node by performing the manual procedure to force the node to rejoin the cluster, documented in Pivotal Knowledge Base.

> **WARNING**: Forcing a node to rejoin the cluster is a destructive procedure. Only perform it with the assistance of Pivotal Support.

It might also be possible to manually dump recent transactions from the detached node and apply them to the running cluster, but this process is error-prone.

Unresponsive Node(s)

Symptom

A MySQL node has become unresponsive.

Explanation

Unresponsive nodes stop responding to queries and, after timing out, leave the cluster.

Nodes are marked as unresponsive or inactive under either of the following circumstances:

- If they fail to respond to one node within 15 seconds
- 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 of the other running nodes and a quorum is held by the remaining nodes.

All nodes suspend writes once they notice a problem with the cluster. After a timeout period of 5 seconds, requests to non-quorum nodes 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 continue fulfilling write requests.

If deployed using a proxy, a continually inactive node causes the proxy to fail over, selecting a different MySQL node to route new queries to.

Solutions

A node can become unresponsive for a number of reasons. Each reason corresponds to a different solution. Consult the following list:

- Network Latency
- MySQL Process Failure
- Firewall Rule Change
- VM Failure
- The Interruptor

Network Latency

If network latency causes a node to become unresponsive, the node drops but eventually rejoins.

The node will only rejoin automatically if only one node has left the cluster.

Consult your IaaS network settings to reduce your network latency.

MySQL Process Failure

If the MySQL process crashes, monit and BOSH will bring the process back. No further action is necessary.

Firewall Rule Change

If firewall rules change, it may prevent a node from reaching the rest of the cluster, causing the node to become unresponsive.

In this scenario, the logs show the node leaving the cluster but do not show network latency errors.

To confirm that the node is unresponsive due to a firewall rule change, try to SSH from a responsive node to the unresponsive node. If you can’t connect, the node is unresponsive due to a firewall rule change.

Change the firewall rules to enable the unresponsive node to rejoin the cluster.

VM Failure

If you cannot SSH onto a node, and you are not detecting either network latency or firewall issues, your node may be down due to VM failure.

To confirm that VM failure has caused the node to become unresponsive, perform the following steps:

1. SSH into the Ops Manager Director. For more information, see the SSH into Ops Manager section of *Advanced Troubleshooting with the BOSH CLI*.
2. Retrieve the IP address for the MySQL server by navigating to the **MySQL for PCF** tile and clicking the **Status** tab.
3. Retrieve the VM credentials for the MySQL server by navigating to the **MySQL for PCF** tile and clicking the **Credentials** tab.
4. From the Ops Manager Director VM, use the BOSH CLI to log in to the BOSH Director. For more information, see the Log in to the BOSH Director section of *Advanced Troubleshooting with the BOSH CLI*.
5. From the Ops Manager VM, use the BOSH CLI to run `bosh cloudcheck`. For more information, see the BOSH Cloudcheck section of *Advanced Troubleshooting with the BOSH CLI*.
6. In `bosh cloudcheck`, identify the unresponsive node and recreate it.

   WARNING: Recreating a node will clear its logs. Ensure the node is completely down before recreating it.

   WARNING: Only recreate one node. Do not recreate the entire cluster. If more than one node is down, contact Pivotal Support.

The Interruptor

If you have enabled the Interruptor, it may be preventing a node from starting.

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)
```

Perform the procedure in the Override the Interruptor section of *Using the Interruptor* to force the node to rejoin the cluster.

Accidental Deletion of Service Plan

Symptom

You have deleted a service plan accidentally.

Explanation

A service plan is only unrecoverable if you perform all of the following steps in sequence:

1. Click the trash-can icon in the **Service Plan** section of the MySQL for PCF configuration.
2. Create a plan with the same name.
3. Click **Save**.
4. Return to the Ops Manager Installation Dashboard, and click **Apply Changes**.

The deploy eventually fails with the following error:

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

Solution

After you have deployed with **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 continue to appear under the old plan name, but new instances need to be created using the new plan name.

If you have performed steps 1 and 2 in the above sequence, do not click **Save**. Return to the Ops Manager Installation Dashboard. Any accidental changes are discarded.

If you have performed steps 1, 2, and 3 in the above sequence, do not click **Apply Changes.** Return to the **Ops Manager Installation Dashboard** and click the **Revert** button. Any accidental changes are discarded.

Bootstrapping

This topic describes how to bootstrap your MySQL cluster in the event of a cluster failure.

When to Bootstrap

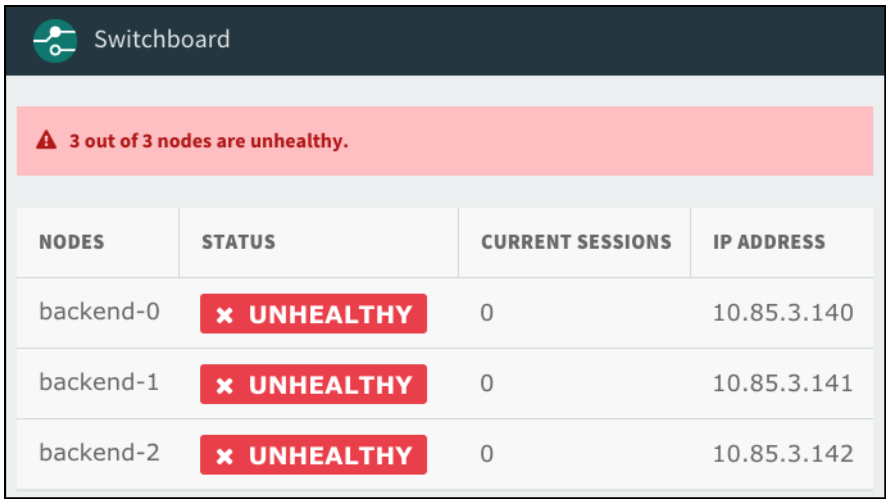
To determine whether you need to bootstrap your cluster, you must check whether the cluster has lost quorum. Bootstrapping is only required when the cluster has lost quorum. See Check Cluster State for more information about checking the state of your cluster.

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, which means the error is resolved, the node is restarted, or connectivity is restored.

To check whether your cluster has lost quorum, look for the following symptoms:

- All nodes appear “Unhealthy” on the proxy dashboard, as in the following screenshot:



- All responsive nodes report the value of `wsrep_cluster_status` as `non-Primary` in the MySQL client.

```
mysql> SHOW STATUS LIKE 'wsrep_cluster_status';
+----------------------+-------------+
| Variable_name        | Value       |
+----------------------+-------------+
| wsrep_cluster_status | non-Primary |
+----------------------+-------------+
```

- All unresponsive nodes respond with `ERROR 1047` when using most statement types in the MySQL client:

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

If your cluster has lost quorum, see the Bootstrapping topic for information about manually bootstrapping a cluster.

Keep in mind the following:

- The start script bootstraps node 0 only on initial deploy. If bootstrapping is necessary at a later date, it must be done manually.
- If the single node is bootstrapped, it creates a new one-node cluster that other nodes can join.

Bootstrapping

Before running the bootstrapping procedures below, you must SSH into the Ops Manager VM and log in to the BOSH Director.

For more information, see Prepare to Use the BOSH CLI.

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

MySQL for PCF v1.8.0 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

In this scenario, the nodes are up and running, but the cluster has been disrupted.

To determine whether the cluster has been disrupted, perform the following steps:

1. Use `bosh vms` to list the jobs and see if they are `failing`. The output will resemble the following:

```
+-----------------------------------------------+---------+-----------------------------------------------+------------+
| 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 |
+-----------------------------------------------+---------+-----------------------------------------------+------------+
```

2. If the jobs are failing, perform the following steps:

    a. Use `bosh deployment` to select the correct deployment.
    b. Use `bosh run errand bootstrap` to run the bootstrap errand.

    You 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)
```

    If the bootstrap errand doesn't work immediately, wait and try it again a few minutes later.

Scenario 2: Virtual Machines Terminated or Lost

In more severe circumstances, such as power failure, all of your VMs might have been lost. You must recreate them before you can begin to recover the cluster. In this scenario, the nodes appear as `unknown/unknown` in the output of `bosh vms`.

Run the BOSH CLI command `bosh vms`. The output will resemble the following:

```
+--------------------------------------------------+--------------------+--------------------------------------------------+------------+
| 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 |
+--------------------------------------------------+--------------------+--------------------------------------------------+------------+
```

Recover Terminated or Lost VMs

To recover your VMs, perform the following steps:

1. If you use the VM Resurrector, disable it.

2. Run the BOSH Cloud Check interactive command. Use the BOSH CLI command `bosh cck`.

   When prompted, select **Recreate VM**. If this option fails, select **Delete VM reference**.

   The output will resemble the following:

```
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]: 2



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]: 2


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]: 2
```

3. Re-enable the VM Resurrector if you want to continue to use it.

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

Update the BOSH Configuration

In a 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.

Perform the following steps:

1. Log in to the BOSH Director.

2. Target the correct deployment.

3. Run `bosh edit deployment`:
   - Locate the `jobs.mysql-partition.update` section.
   - Change `max_in_flight` to `3`.

- Below the `max_in_flight` line, add a line: `canaries: 0`.

4. Run `bosh deploy`.

Run the Bootstrap Errand

1. Run the BOSH CLI command `bosh run errand bootstrap`.
2. Validate that the errand completes successfully. Even if some instances still appear as `failing`, proceed to the next step.

Restore the BOSH Configuration

To restore your BOSH configuration to its previous state, follow the previous steps from Update the BOSH Configuration and use the following values:

- Set `canaries` to `1`.
- Set `max_in_flight` to `1`.
- Set `serial` to `true` in the same manner as above.

After updating the BOSH configuration, perform the following steps:

1. Redeploy your BOSH Director.
2. Validate that all `mysql` instances are in `running` state.

Note: You must run all of the steps. If you do not re-set the values in the BOSH manifest, the status of the jobs is not reported correctly and can lead to problems with future deploys.

Manual Bootstrap

If the bootstrap errand is not able to automatically recover the cluster, you might need to perform the steps manually.

WARNING: The following procedures are prone to user-error and can result in lost data if followed incorrectly. Follow the assisted bootstrap procedure in Bootstrapping above first, and only resort to the manual process if the errand fails to repair the cluster.

Perform the procedures in the sections below to manually bootstrap your cluster.

Shut Down MariaDB

Perform the following steps for each node in the cluster:

1. SSH into the node. For more information about how to SSH into BOSH-deployed VMs, see BOSH SSH.
2. Shut down the `mariadb` process on the node. Run the following command:

```
monit stop mariadb_ctrl
```

Re-bootstrapping the cluster is not successful unless you shut down the `mariadb` process on all nodes in the cluster.

Choose Node to Bootstrap

To choose the node to bootstrap, you must find the node with the highest transaction sequence number (`seqno`).

Perform the following steps to find the node with the highest `seqno`:

1. Run the following command from the node:

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

2. If a node shut down gracefully, the `seqno` is in the Galera state file. Retrieve the `seqno` and continue to Bootstrap the First Node.

   If a node crashed or was killed, the `seqno` in the Galera state file is recorded as `-1`. In this case, the `seqno` might be recoverable from the database. Run the following command to start up the database, log the recovered `seqno`, and then exit:

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

   Scan the error log for the recovered `seqno`. It is the last number after the group id (`uuid`). For example:

   ```
   $ grep "Recovered position" /var/vcap/sys/log/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 is nothing to recover. Unless all nodes crashed this way, do not choose this node for bootstrapping.

3. After determining the `seqno` for all nodes in your cluster, identify the node with the highest `seqno`. If all nodes have the same `seqno`, you can choose any node as the new bootstrap node.

Bootstrap the First Node

After determining the node with the highest `seqno`, perform the following steps to bootstrap the node:

> **Note:** Only perform these bootstrap commands on the node with the highest `seqno`. Otherwise the node with the highest `seqno` is unable to join the new cluster unless its data is abandoned. Its `mariadb` process will exit with an error. For more information about intentionally abandoning data, see Architecture.

1. On the new bootstrap node, update the state file and restart the `mariadb` process. Run the following commands:

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

2. It can take up to ten minutes for `monit` to start the `mariadb` process. To check if the `mariadb` process has started successfully, run the following command:

   ```
   watch monit summary
   ```

Restart Remaining Nodes

1. After the bootstrapped node is running, start the `mariadb` process on the remaining nodes with `monit`. From the bootstrap node, run the following command:

   ```
   monit start mariadb_ctrl
   ```

   If the node is prevented from starting by the Interruptor, perform the manual procedure to force the node to rejoin the cluster, documented in Pivotal Knowledge Base.

   > **WARNING:** Forcing a node to rejoin the cluster is a destructive procedure. Only perform it with the assistance of Pivotal Support.

2. If the `monit start` command fails, it might be because the node with the highest `seqno` is `mysql/0`. In this case, perform the following steps:

   a. From the Ops Manager VM, run the following command to make BOSH ignore updating `mysql/0`:

      ```
      bosh ignore mysql/0
      ```

   b. Navigate to Ops Manager in a browser, log in, and click **Apply Changes**.
   c. When the deploy finishes, run the following command from the Ops Manager VM:

      ```
      bosh unignore mysql/0
      ```

3. Verify that the new nodes have successfully joined the cluster. SSH into the bootstrap node and run the following command to output the total number of nodes in the cluster:

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

Using the Interruptor

This topic explains how to use the Interruptor, a component of MySQL for Pivotal Cloud Foundry (PCF) that provides a solution for preventing data loss.

WARNING: Using the Interruptor is potentially destructive and should only be attempted by advanced users.

Overview

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 the Interruptor 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, 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 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 might invoke SST. In cases of failed replication, SST can cause data loss. When enabled, the Interruptor prevents this method of recovery.

For more information about these modes, see State Transfers in the Galera documentation.

Sample Notification E-mail

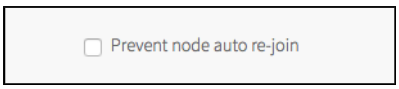
The Interruptor sends an email through the Elastic Runtime notification service when it prevents a node from rejoining a cluster. See the following example:

```
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.
```

Enable the Interruptor

The Interruptor is deactivated by default. To enable it, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the MySQL for PCF tile.
3. Click **Advanced Options**.



4. Under **Enable optional protections**, select the **Prevent node auto re-join** checkbox.
5. Click **Save**.
6. Return to the Ops Manager Installation Dashboard and click **Apply Changes** to redeploy the tile.

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)
```

Override the Interruptor

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 Check Replication Status instructions.

Before running the BOSH CLI commands below, you must SSH into the Ops Manager VM and log in to the BOSH Director. For more information, see Advanced Troubleshooting with the BOSH CLI.

If you are using PCF v1.10, follow the BOSH CLI v1 procedures to force a node to rejoin the cluster.

If you are using PCF v1.11 or later, follow the BOSH CLI v2 procedures to force a node to rejoin the cluster.

Force a Node to Rejoin the Cluster with the BOSH CLI v1

1. List your deployments:

```
$ bosh deployments
```

2. From the output, locate the MySQL for PCF deployment and record its name.
3. Download the manifest for the MySQL for PCF deployment, specifying the name of the deployment. For example:

```
$ bosh download manifest p-mysql-deployment ./manifest.yml
```

4. Set the BOSH CLI to the MySQL for PCF deployment:

```
$ bosh deployment ./manifest.yml
```

5. 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)
```

If the `rejoin-unsafe` errand is not able to cause a node to join the cluster, log in to each node that has tripped the Interruptor and perform the manual procedure to force the node to rejoin the cluster, documented in Pivotal Knowledge Base.

WARNING: Forcing a node to rejoin the cluster is a destructive procedure. Only perform it with the assistance of Pivotal Support.

Force a Node to Rejoin the Cluster with the BOSH CLI v2

1. List your deployments. For example:

    ```
    $ bosh2 -e my-env deployments
    ```

2. From the output, locate the MySQL for PCF deployment and record its name.

3. Run `bosh2 run-errand rejoin-unsafe` to force a node to rejoin the cluster, specifying the MySQL for PCF deployment. For example:

    ```
    $ bosh2 -e my-env -d p-mysql-deployment 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)
    ```

If the `rejoin-unsafe` errand is not able to cause a node to join the cluster, log in to each node that has tripped the Interruptor and perform the manual procedure to force the node to rejoin the cluster, documented in Pivotal Knowledge Base.

WARNING: Forcing a node to rejoin the cluster is a destructive procedure. Only perform it with the assistance of Pivotal Support.

Rotating MySQL for PCF Credentials

Page last updated:

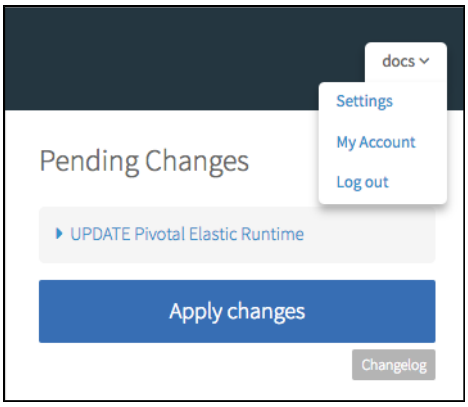
This topic describes how to rotate credentials for MySQL for Pivotal Cloud Foundry (MySQL for PCF). If you are also using Elastic Runtime MySQL, review the notes in this procedure in order to rotate credentials for both products.

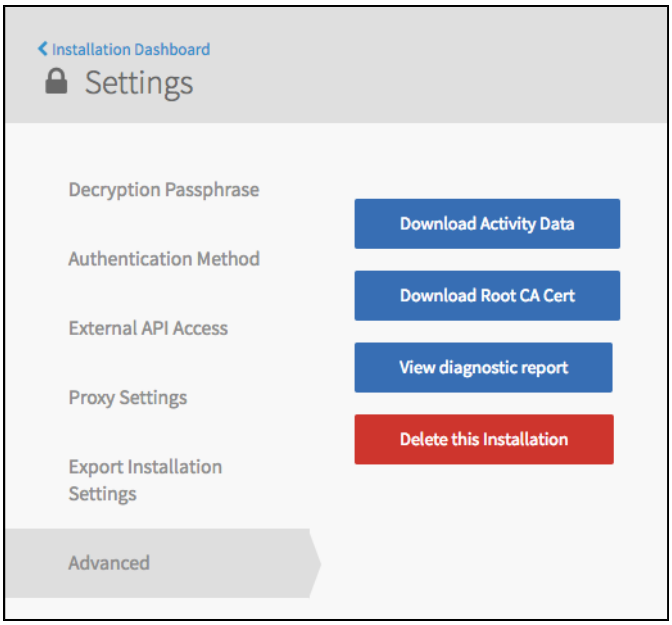
Prerequisites

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

1. Your root CA certificate in a `.crt` 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**.

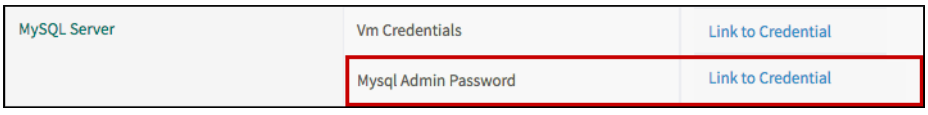
    

    b. Click **Advanced**.

    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`.

> **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: 8419c793d377429aa40eea07fb6e7686
    ```

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 v1.9.0 and later, the `mysql-diag` tool is automatically installed and configured. If you are on a prior version, select the documentation for your version of MySQL for PCF.

Run mysql-diag

1. If this is your first time using `mysql-diag`, follow the instructions within the Prepare to Use the BOSH CLI topic. If you have completed this step in a prior use of `mysql-diag`, move directly to Step 2.
2. Select either Elastic Runtime or MySQL for PCF as your deployment to troubleshoot. Perform the steps detailed in the Select a Product Deployment to Troubleshoot topic.
3. Run `bosh ssh` from the Ops Manager VM and select the **mysql-monitor node** from the list. Perform the steps detailed in the Select a Product Deployment to Troubleshoot topic.
4. Once on the `mysql-monitor` VM type the following command to run `mysql-diag`:

```
$ mysql-diag
```

mysql-diag-agent

MySQL for PCF v1.9.0 and later will have the `mysql-diag-agent` present. Older versions 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

The `mysql-diag` command returns the following message if your canary status is healthy:

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

Here is a sample `mysql-diag` output after the tool identified a healthy cluster:

```
mysql-monitor/5faae86a-a325-44fa-a8ab-24e736e42390:~$ mysql-diag
Wed Jul 19 19:03:54 UTC 2017
Checking canary status...
Checking canary status... healthy
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... done
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... done
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... done
+------------+---------------+------------------+--------------------+--------------------+
|    HOST    |   NAME/UUID   | WSREP LOCAL STATE | WSREP CLUSTER STATUS | WSREP CLUSTER SIZE |
+------------+---------------+------------------+--------------------+--------------------+
| 10.244.7.2 | mysql/32665833 | Synced           | Primary            |                  3 |
| 10.244.8.2 | mysql/026d5b79 | Synced           | Primary            |                  3 |
| 10.244.9.2 | mysql/fefced1e | Synced           | Primary            |                  3 |
+------------+---------------+------------------+--------------------+--------------------+
I don't think bootstrap is necessary
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... done
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... done
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... done
+------------+---------------+-----------------------------------+-----------------------------------+
|    HOST    |   NAME/UUID   |        PERSISTENT DISK USED       |          EPHEMERAL DISK USED      |
+------------+---------------+-----------------------------------+-----------------------------------+
| 10.244.8.2 | mysql/026d5b79 | 28.9% of 9G (0.0% of 0.61M inodes) | 28.3% of 157.4G (6.4% of 10.49M inodes) |
| 10.244.7.2 | mysql/32665833 | 28.7% of 9G (0.0% of 0.61M inodes) | 28.3% of 157.4G (6.4% of 10.49M inodes) |
| 10.244.9.2 | mysql/fefced1e | 28.9% of 9G (0.0% of 0.61M inodes) | 28.3% of 157.4G (6.4% of 10.49M inodes) |
+------------+---------------+-----------------------------------+-----------------------------------+
```

View a larger version of this image.

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 that cluster. Below is a sample `mysql-diag` output after the tool identified an unhealthy cluster:

```
mysql-monitor/5faae86a-a325-44fa-a8ab-24e736e42390:~$ mysql-diag
Wed Jul 19 17:59:43 UTC 2017
Checking canary status...
Checking canary status... healthy
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... dial tcp 10.244.9.2:3306: getsockopt: connection refused
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... dial tcp 10.244.7.2:3306: getsockopt: connection refused
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... dial tcp 10.244.8.2:3306: getsockopt: connection refused
+------------+---------------+------------------+--------------------+-------------------+
|    HOST    |   NAME/UUID   | WSREP LOCAL STATE | WSREP CLUSTER STATUS | WSREP CLUSTER SIZE |
+------------+---------------+------------------+--------------------+-------------------+
| 10.244.9.2 | mysql/fefced1e | N/A - ERROR       | N/A - ERROR          | N/A - ERROR       |
| 10.244.7.2 | mysql/32665833 | N/A - ERROR       | N/A - ERROR          | N/A - ERROR       |
| 10.244.8.2 | mysql/026d5b79 | N/A - ERROR       | N/A - ERROR          | N/A - ERROR       |
+------------+---------------+------------------+--------------------+-------------------+
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... done
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... done
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... done
+------------+---------------+-----------------------------------+----------------------------------------+
|    HOST    |   NAME/UUID   |        PERSISTENT DISK USED       |           EPHEMERAL DISK USED          |
+------------+---------------+-----------------------------------+----------------------------------------+
| 10.244.7.2 | mysql/32665833 | 28.7% of 9G (0.0% of 0.61M inodes) | 28.2% of 157.4G (6.3% of 10.49M inodes) |
| 10.244.8.2 | mysql/026d5b79 | 28.9% of 9G (0.0% of 0.61M inodes) | 28.2% of 157.4G (6.3% of 10.49M inodes) |
| 10.244.9.2 | mysql/fefced1e | 28.9% of 9G (0.0% of 0.61M inodes) | 28.2% of 157.4G (6.3% of 10.49M inodes) |
+------------+---------------+-----------------------------------+----------------------------------------+

[CRITICAL] You must bootstrap the cluster. Follow these instructions: http://docs.pivotal.io/p-mysql/bootstrapping.html

[CRITICAL] Run the download-logs command:
$ download-logs -d /tmp/output -n 10.244.7.2 -n 10.244.8.2 -n 10.244.9.2
For full information about how to download and use the download-logs command see https://discuss.pivotal.io/hc/en-us/articles/221504408

[WARNING] NOT RECOMMENDED
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.
```

View a larger version of this image.

Getting Started with PCF MySQL Galera

This topic explains how a developer on PCF can start using MySQL with their apps. It is important to note that MySQL for PCF v1 is different than standard MySQL and has some limitations that you must be aware of.

MySQL for PCF v1 Limitations

- Only InnoDB tables are supported. Writes to other types of tables, such as MyISAM tables will not replicate across the cluster.
- Explicit locking is not supported, i.e. `LOCK TABLES` , `FLUSH TABLES tableA WITH READ_LOCK` .
- Large DDL (ie, schema changes like ALTER TABLE) will lock all schemas, affecting all sessions with the DB. This can be mitigated via a manual step using Galera's RSU feature.
- Table partitioning may cause the cluster to get into a hung state. This is as a result of the implicit table locks that are used when running table partition commands.
- MySQL for PCF supports table triggers; however multiple triggers per table are not supported
- All tables must have a primary key; multi-column primary keys are OK. This is because of the the way Galera replicates using row based replication and ensuring unique rows on each instance
- While not explicitly a limitation, large transaction sizes may inhibit the performance of the cluster and thus the applications using the cluster. In a MariaDB Galera cluster, writes are processed as "a single memory-resident buffer", so very large transactions will adversely affect cluster performance.
- Do not execute a DML statement in parallel with a DDL statement when both statements affect the same tables. Locking is lax in Galera, even in single node mode. Rather than the DDL waiting for the DML to finish, they will both apply immediately to the cluster and may cause unexpected side effects .
- Do not rely on auto increment values being sequential as Galera guarantees auto-incrementing unique non-conflicting sequences, so each node will have gaps in IDs. Furthermore, Galera sets user's to READ ONLY in regards to auto increment variables. Without this feature, Galera would require shared locking of the auto increment variables across the cluster, causing it to be slower and less reliable
- MySQL for PCF does not support MySQL 5.7's JSON
- Max size of a DDL or DML is limted to 2GB
- Defining whether the node splits large `Load Data` commands into more maneageable units

Check Your App for Limitations

Certain types of queries may cause deadlocks. For example, transactions like `UPDATE` or `SELECT ... for UPDATE` when querying rows in opposite order will cause the queries to deadlock. Rewriting these queries and SQL statements will help minimize the deadlocks that your application experiences. One such solution is to query for a bunch of potential rows, then do an update statement. The MySQL documentation provides more information about InnoDB Deadlocks and Handling InnoDB Deadlocks.

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 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.

MySQL for PCF Tools

The following tools let developers check the usage of their MySQL for PCF service instances, and access their databases directly.

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
     ```

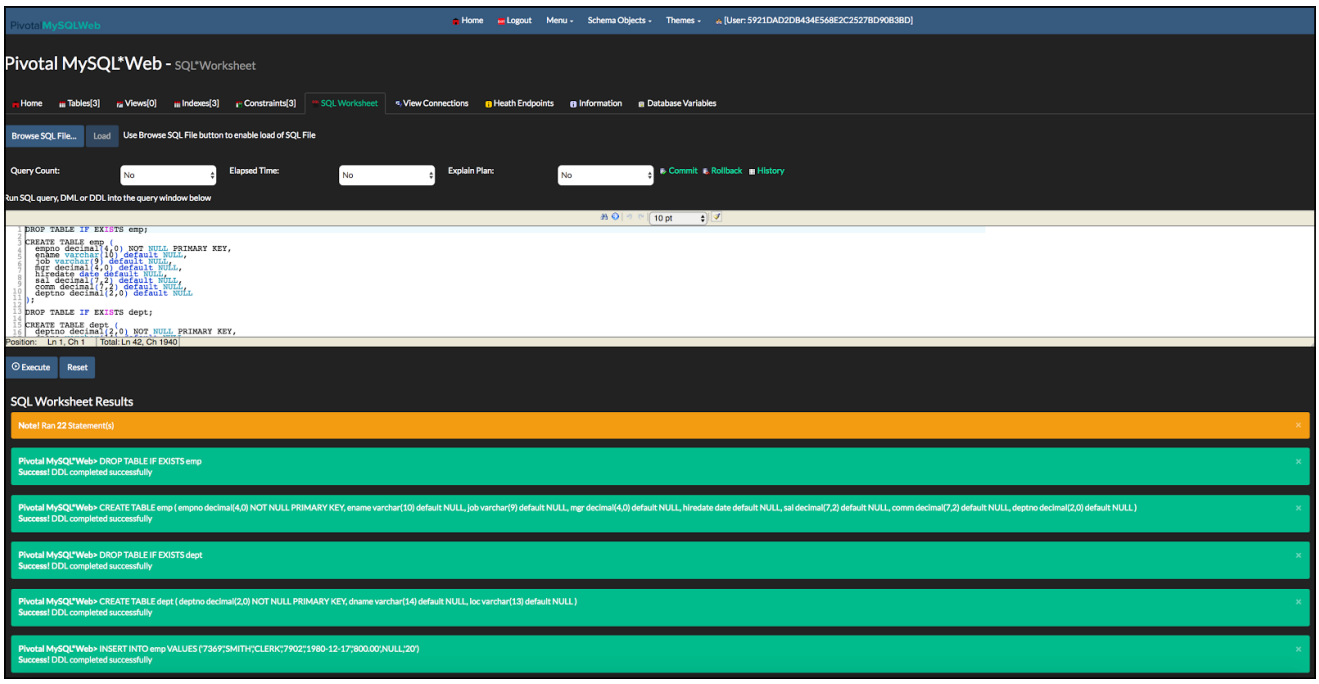
  3. Navigate to the URL listed in the output as `Dashboard`. In the example above, the instance dashboard URL is `https://p-mysql.sys.acceptance.cf-app.example.com/manage/instances/ddfa6842-b308-4983-a544-50b3d1fb62f0`.

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.

When to use RSU for a DDL

RSU or Rolling Schema Upgrade is a method of performing schema upgrades in which an operator manually takes each node in the cluster out of the cluster and applies the DDL by hand. Because these are happening to each node in turn, the DDL must be backwards-compatible with the previous schema.

Some DDLs like DROP statements are fast and do not require the oversight of an operator. Other DDLs, like adding an INDEX to a large table can take a very long time to run. While they are running, the cluster will seem unavailable to anyone trying to use it. RSU allows taking a node out of the cluster so that it doesn't block during these operations.

We suggest seeking help to perform a DDL via RSU in the event that you are doing large and long running changes to the database schema. It is hard to know exactly how long a given migration will take. Try to run the migrations on a test environment first and see how long they take as an approximation to gauge how long the production DDL will take. Then, if this is longer than the cluster can be "unavailable" consult with your operator about using RSU for DDL.

Note: If you have a DDL of >2GB, you must use RSU as it will always fail under TOI.

Steps for Performing an RSU DDL

The following steps assume that you are using import sql file to perform the data imports. In the event you choose to execute a DDL manually with the mysql client, run the following steps using the client in order.

1. Import base table structure and data in normal TOI mode. At this point, all MySQL nodes should have the same schema structure
   `$ mysql -u<USER> -p -h<IP ADDRESS FOR A MYSQL NODE> -D<DATABASE_NAME> < path/to/schema_import_file.sql`

2. Do the following for each node as the admin user:

   a. Enable global read_only and verify that the node is unhealthy via the proxies by doing the following:

      i. `$ mysql -u<USER> -p -h<MYSQL HOST> -e "set global read_only=on;"`
      ii. Find the values of `cf_mysql.proxy.api_username` and `cf_mysql.proxy.api_password`
      iii. Visit 0-proxy-p-mysql.SYSTEM_DOMAIN and login with the above credentials
      iv. One of the backends should show up as a red bar that says "unhealthy"

   b. Add `set wsrep_osu_method=rsu;` to the top of the import file containing the constraints and indexes
   c. Add `set global wsrep_desync=on;` to the top of the constraints and indexes file. Add `set global wsrep_desync=off;` to the bottom of the constraints and indexes file.
   d. At this point your file should look like
      `$ cat constraints_and_indexes.sql set wsrep_osu_method=rsu; set global wsrep_desync=on; [the previous file contents] set global wsrep_desync=off;`
   e. Execute the constraints and index modification file
      `$ mysql -u<USER> -p -h<MYSQL HOST> -D<DATABASE_NAME> < path/to/constraint_and_index_file.sql`
   f. Disable global read_only and verify that all nodes are healthy via the proxies

      i. `$ mysql -uUSER -p -h<MYSQL HOST> -e "set global read_only=off;"`
      ii. Visit 0-proxy-p-mysql.SYSTEM_DOMAIN and login with the above credentials
      iii. All of the backends should show up with a green bar that say healthy

   g. Repeat for the remaining nodes

3. At this point your cluster should be stable and running with the new schema