

MySQL for PCF®

Documentation

Version 2.2

Published: 9 May 2019

Table of Contents

Table of Contents	2
MySQL for PCF v2	3
Overview of MySQL for PCF v2.2	5
Release Notes	7
On-Demand Service Architecture	9
About Leader-Follower	11
MySQL for PCF Recommended Usage and Limitations	13
Installing and Configuring MySQL for PCF	15
Backing Up and Restoring On-Demand MySQL for PCF	25
Setting Limits for On-Demand Service Instances	36
Controlling Access to Service Plans by Org	40
Monitoring and KPIs for On-Demand MySQL for PCF	41
Using Leader-Follower Topology for Availability	46
Events that Interrupt Service	52
Upgrading MySQL for PCF	54
Troubleshooting MySQL for PCF	56
Using MySQL for PCF v2	75
MySQL Server Defaults	80
Customizing Database Credentials	83
Migrating Data from MySQL for PCF v1 to v2	85
MySQL Environment Variables	88
Troubleshooting On-Demand Instances	89

MySQL for PCF v2

Note: MySQL for PCF v2.2 is no longer supported because it has reached the End of General Support phase. To stay up-to-date with the latest software and security updates, upgrade to a supported version.

This guide describes setting up and using MySQL for Pivotal Cloud Foundry (PCF) as an on-demand service.

Note: This documents MySQL for PCF v2, an on-demand service that creates dedicated, single-tenant service instances dynamically. For the older, pre-provisioned service, see the v1.10 [documentation](#) or select **v1.10** from the drop-down menu above.

About MySQL for PCF

MySQL for PCF enables PCF app developers to provision and use a MySQL database with a single command.

The MySQL for PCF product delivers dedicated instances on demand, “Database as a Service”, to PCF users. When installed, the tile deploys and maintains a single Service Broker that is responsible for PCF integration. The service is configured with sane defaults, following the principle of least surprise for a general-use relational database service.

For more information about MySQL for PCF v2.2, see [Minor Overview](#).

MySQL for PCF v2 introduces a new type of service, an *on-demand* service. MySQL for PCF v2 offers an alternative to the *pre-provisioned* service that was in the v1.x series.

This table summarizes the main differences between the two:

	Available Since	VMs it Runs On	How VMs are Created	Metrics Name Prefix
On-Demand Service	New for v2	Dedicated instance VM(s) dynamically provisioned on demand.	PCF creates each VM on-demand when app developer creates service instance	<code>p.mysql</code> (with a dot)
Pre-Provisioned Service	v1.4	Multi-tenant VMs shared by apps across PCF deployment	PCF creates all VMs when operator deploys or updates service	<code>p-mysql</code> (with a dash)

Product Snapshot

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

Element	Details
Version	v2.2.7
Release date	November 29, 2018
Software component version	Percona Server v5.7.20-21
Compatible Ops Manager version(s)	v1.12.x and v2.0.x
Compatible Elastic Runtime* version(s)	v1.12.0
Compatible Pivotal Application Service (PAS)* version(s)	v2.0.x
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere
IPsec support	Yes

* As of PCF v2.0, *Elastic Runtime* is renamed *Pivotal Application Service (PAS)*. For more information, see [Pivotal Application Service \(PAS\) Highlights](#).

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 and MySQL for PCF, only on-demand service plans are available.

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	NA
PCC	Pivotal GemFire	All versions	NA

Feedback

Please provide any bugs, feature requests, or questions to [the Pivotal Cloud Foundry Feedback list](#).

Overview of MySQL for PCF v2.2

This topic describes the significant new features in MySQL for PCF v2.2. This topic also presents a checklist that you can use to decide if MySQL for PCF is ready to meet your business requirements.

Introduction

MySQL for PCF v2 introduces the On-Demand Service, which provides dedicated instances of MySQL that App Developers can provision on-demand from the command line. The MySQL for PCF On-Demand service is designed to improve the flexibility in how and when instances are provisioned and allows for more efficient and seamless resource use for both Operator and App Developer. Additionally, both Operator and App Developer can configure essential MySQL settings. MySQL for PCF v2 can be installed alongside the earlier MySQL for PCF v1.x tiles.

MySQL for PCF's On-Demand Service leverages the [on-demand service broker](#).

On Demand Service Plan Descriptions

PCF MySQL offers five on-demand plans as the `p.mysql` service within the PCF MySQL tile. The plans can be configured in any way that you see fit. However, plans should be configured such that they increase in size.


For each service plan, the operator can configure the **Multi-Node**, **Plan name**, **Plan description**, **Server VM type** and **Server Disk type**, or choose to disable the plan completely.

Operators can update certain plan settings after the plans have been created. If the Operator updates the VM size or disk size these settings will be implemented in all instances already created. Operators should not downsize the VMs or disk size as this can cause data loss in pre-existing instances.

Enterprise-Ready Checklist

Review the following table to determine if MySQL for PCF v2 has the features needed to support your enterprise.

Plans and Instances		More Information
On-Demand, Dedicated-VM plans	MySQL for PCF provides On-Demand VM plans	Architecture
Customizable plans	For the On-Demand Plan, the operator can customize the VM, disk size, and availability zone.	Configuring
Installation and Upgrades		More Information
Product upgrades	MySQL for PCF can be upgraded within the v2.x tile series. Additionally, it can be installed alongside the MySQL for PCF v1.X tiles to enable easy manual migrations	Upgrading MySQL for PCF
Deployment Smoke Tests	MySQL for PCF installation and upgrade runs a post deployment BOSH errand that validates basic MySQL operations	
Maintenance and Backups		More Information
Operational Monitoring and Logging	MySQL for PCF v2 provides metrics for monitoring On-Demand plan usage and quotas, as well as MySQL component metrics. Additionally, MySQL for PCF provides syslog redirection to external log ingestors.	Monitoring MySQL for PCF
Backup and Restore	MySQL for PCF v2 includes automatic backups on a configurable schedule to a variety of destinations, as well as a simple restore process.	Manual Backup and Restore of MySQL for PCF
Scale and Availability		More Information
On-Demand Plan	MySQL for PCF provides up to 50 on-demand instances across all plans	
Ability to Scale Up / Down	Operators can scale VMs up, but not down	Configuring
Rolling Deployments	MySQL for PCF does not support rolling deployments because it is single node; the service is unavailable during upgrades.	Upgrades
AZ Support	MySQL for PCF can be deployed to multiple zones to increase availability during unplanned outages of a zone.	About Multiple AZs in MySQL for PCF

Encryption		More Information
Encrypted Communication in Transit	MySQL for PCF has been tested successfully with the BOSH IPsec Add-on	Securing Data in Transit with the IPsec Add-on 

About Multiple AZs in MySQL for PCF v2

MySQL for PCF v2 supports configuring multiple AZs. However, assigning multiple AZs to MySQL jobs does not guarantee high availability.

- On-Demand plans can be assigned to any of the configured availability zones.
- You can select two AZs for the multi-node plans to increase the availability. Choosing three AZs for multi-node plans does not increase the number of AZs assigned to three.
- You can select one AZ for the single-node plan. Choosing two AZs for the single-node plan does not increase the number of AZs assigned to two.

Release Notes

Pivotal recommends that you upgrade to the latest version of your current minor line, then upgrade to the latest available version of the new minor line. For example, if you use an older v2.0.x version, upgrade to the latest v2.0.x version before upgrading to the latest v2.1.x version.

For product versions and upgrade paths, see the [Product Compatibility Matrix](#).

v2.2.7

Release Date: November 29, 2018

Security Fixes

This release includes the following security fix:

- Critical [CVE-2018-15759: On Demand Services SDK Timing Attack Vulnerability](#)

v2.2.5

Release Date: May 8, 2018

- Fixes an issue where updating a service instance may hang indefinitely after MySQL crashes.
- MySQL user for collecting metrics scoped to only connect over localhost.
- Prevents users from creating a binding specifying a custom username of an existing user.
- Requires stemcell 3445.42.

v2.2.4

Release Date: March 15, 2018

- Removes the colocated errands feature added in v2.2.0. Errands now run in individual errand VMs, the same as in v2.1.
- Upgrades Percona Server to 5.7.20-21.
- Requires stemcell 3445.28.

v2.2.3

Release Date: February 16, 2018

- Fixes a service-binding issue, where the service broker returned the IP address of the original leader.
- Minor usability improvements to leader-follower orchestration errands.

v2.2.2

Release Date: February 8, 2018

- When configuring backups to an S3-compatible endpoint, **region** is no longer a required property.
- Requires stemcell 3445.24.

v2.2.1

Release Date: January 23, 2018

- Prevents orphaning a disk when updating a service instance to a plan in a different AZ.
- Upgrades Percona Server to v5.7.20-19.
- Requires stemcell 3445.23.

v2.2.0

Release Date: January 23, 2018

- MySQL for PCF v2.2 requires PCF v1.12 or later or PCF v2.0 or later.
- Offers the ability to create multi-AZ leader-follower instances. Operators can monitor leader-follower instances with replication metrics. For more information, see [About Leader-Follower](#).
- No longer includes the option to disable automated backups. You must configure automated backups. For more information, see [Backing Up and Restoring On-Demand MySQL for PCF](#).
- Updates the mysql-restore utility to make restoring from a backup artifact easier. For more information, see [Restore the Service Instance](#).
- Adds the ability to enable create service binding and service keys with custom usernames. Also introduces **read only** bindings. For more information, see [Customizing Database Credentials](#).
- Enables developers to use workload profiles for best practices in server configuration. For more information, see [Changing MySQL Server Defaults](#).
- No longer includes individual errand VMs. Instead, errands are run on the MySQL VMs. These colocated errands run faster and use fewer resources.
- Upgrades several dependencies including, `golang 1.9.1`.
- Upgrades Percona Server to v5.7.20-18.
- Requires stemcell 3445.19.

On-Demand Service Architecture

This topic describes the architecture for on-demand MySQL for Pivotal Cloud Foundry (PCF).

For information about architecture of the older, pre-provisioned service, see the [Architecture](#) topic for MySQL for PCF v1.9.

Service Network Requirement

When you deploy PCF, you must create a statically defined network to host the component virtual machines that constitute the PCF infrastructure.

PCF components, like the Cloud Controller and UAA, run on this infrastructure network. In PCF v2.0 and earlier, on-demand PCF services require that you host them on a network that runs separately from this network.

Cloud operators pre-provision service instances from Ops Manager. Then, for each service, Ops Manager allocates and recovers static IP addresses from a pre-defined block of addresses.

To enable on-demand services in PCF v2.0 and earlier, operators must create a service networks in BOSH Director and select the **Service Network** checkbox. Operators then can select the service network to host on-demand service instances when they configure the tile for that service.

Default Network and Service Network

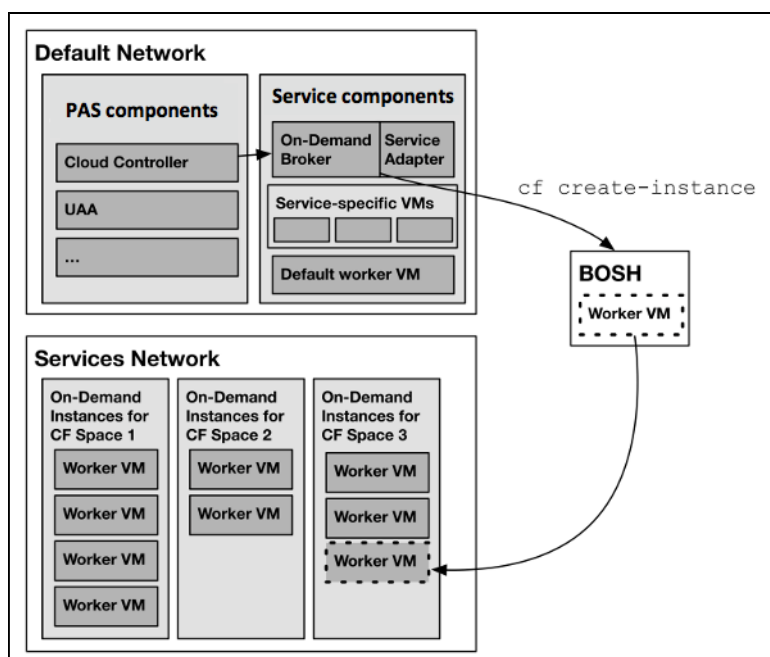
On-demand PCF services rely on the BOSH 2.0 ability to dynamically deploy VMs in a dedicated network. The on-demand service broker uses this capability to create single-tenant service instances in a dedicated service network.

On-demand services use the dynamically-provisioned service network to host the single-tenant worker VMs that run as service instances within development spaces. This architecture lets developers provision IaaS resources for their service instances at creation time, rather than the operator pre-provisioning a fixed quantity of IaaS resources when they deploy the service broker.

By making services single-tenant, where each instance runs on a dedicated VM rather than sharing VMs with unrelated processes, on-demand services eliminate the “noisy neighbor” problem when one app hogs resources on a shared cluster. Single-tenant services can also support regulatory compliance where sensitive data must be compartmentalized across separate machines.

An on-demand service splits its operations between the default network and the service network. Shared components of the service, such as executive controllers and databases, run centrally on the default network along with the Cloud Controller, UAA, and other PCF components. The worker pool deployed to specific spaces runs on the service network.

The diagram below shows worker VMs in an on-demand service instance running on a separate services network, while other components run on the default network.



Required Networking Rules for On-Demand Services

Before deploying a service tile that uses the on-demand service broker (ODB), request the needed network connections to allow components of Pivotal Cloud Foundry (PCF) to communicate with ODB.

The specifics of how to open those connections varies for each IaaS.

See the following table for key components and their responsibilities in an on-demand architecture.

Key Components	Their Responsibilities
BOSH Director	Creates and updates service instances as instructed by ODB.
BOSH Agent	Includes an agent on every VM that it deploys. The agent listens for instructions from the BOSH Director and carries out those instructions. The agent receives job specifications from the BOSH Director and uses them to assign a role, or job, to the VM.
BOSH UAA	Issues OAuth2 tokens for clients to use when they act on behalf of BOSH users.
PAS	Contains the apps that are consuming services
ODB	Instructs BOSH to create and update services, and connects to services to create bindings.
Deployed service instance	Runs the given data service. For example, the deployed Redis for PCF service instance runs the Redis for PCF data service.

Regardless of the specific network layout, the operator must ensure network rules are set up so that connections are open as described in the table below.

This component...	Must communicate with...	Default TCP Port	Communication direction(s)	Notes
BOSH Agent	BOSH Director	4222	Two-way	The BOSH Agent runs on every VM in the system, including the BOSH Director VM. The BOSH Agent initiates the connection with the BOSH Director. The default port is not configurable.
Broker and service instances	the Doppler on PAS or Elastic Runtime	8082	One-way	This port is for metrics.
Deployed apps on PAS or Elastic Runtime	MySQL service instances	3306	One-way	This port is for general use, app-specific tasks. In addition to configuring your IaaS, create a security group for the MySQL service instance.
Leader VM	Follower VM	8443 8081	Two-way	This port is needed if leader-follower is enabled. For information, see Configure a Leader-Follower Service Plan .
ODB	<ul style="list-style-type: none"> BOSH Director BOSH UAA 	<ul style="list-style-type: none"> 25555 8443 	One-way	The default ports are not configurable.
ODB	MySQL service instances	3306	One-way	This connection is for administrative tasks. Avoid opening general use, app-specific ports for this connection.
ODB	PAS or Elastic Runtime	8443	One-way	The default port is not configurable.
PAS or Elastic Runtime	ODB	8080	One-way	This port allows PAS or Elastic Runtime to communicate with the ODB component.

About Leader-Follower

This topic describes how the leader-follower topology works in MySQL for Pivotal Cloud Foundry (PCF) and contains information to help users decide whether to enable or use leader-follower.


Understanding Leader-Follower

The leader-follower topology increases the availability of a MySQL database by deploying two MySQL VMs in two separate availability zones (AZs). When a developer creates a leader-follower service instance, the on-demand broker deploys a leader VM in one AZ, and a follower VM in another AZ. Any data that is written to the leader is asynchronously replicated to the follower.

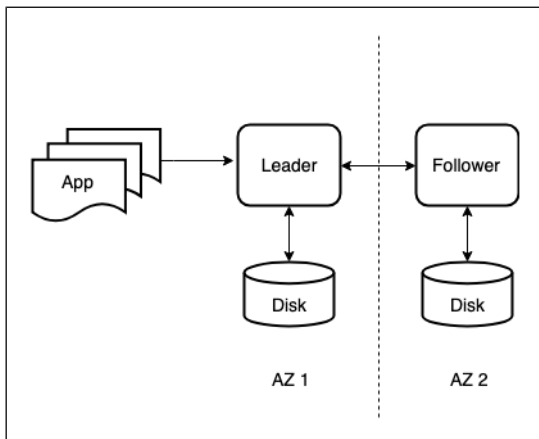
A leader-follower topology enables operators to initiate a failover and send app traffic to the follower if the leader fails. This ensures that the app bound to the MySQL database continues to function normally.

The follower VM is only for increasing availability and is not exposed to developers to increase read throughput. Developers who want increased read throughput can do this by configuring `workload` profiles. For information, see [Understanding Workload Types](#).

For high-level information on the leader-follower process, see [Leader-Follower Setup and Failover Process](#).

 **Note:** The term leader-follower is analogous to master-slave.

The following diagram shows a leader-follower service instance in two availability zones (AZs):



Infrastructure Requirements for Leader-Follower Deployments

Leader-follower instances have additional infrastructure requirements to singleton instances, as described below.

Capacity Planning

When calculating IaaS usage, you must take into account that each leader-follower instance requires two VMs. Therefore, the resources used for a leader-follower-enabled plan must be doubled. For more information, see [Resource Usage Planning for On-Demand Plans](#).

Availability Zones

To minimize impact of an AZ outage and to remove single points of failure, Pivotal recommends that you provision three AZs if using leader-follower deployments. With three AZs, the MySQL VMs are deployed to two AZs, and the broker is deployed to a third.

Networking Rules

In addition to the standard networking rules needed for MySQL for PCF, the operator must ensure leader-follower-specific network rules are also set up as

follows:

- Leader-follower VMs bidirectionally communicate with each other over port 8008 for orchestration.
- Leader-follower VMs bidirectionally communicate with each other over port 3306 for replication.

For information about the standard networking rules, see [Required Networking Rules for On-Demand Services](#).

Enabling and Using Leader-Follower

For information on enabling and using leader-follower, see [Using Leader-Follower Topology For Availability](#).

Understanding Failover

MySQL for PCF focuses on data consistency rather than availability. Therefore, it does not trigger failover automatically, but relies on operators to trigger a failover. When an operator triggers a failover, app traffic is sent to the follower.

For more information, see [Trigger a Failover](#).

Understanding Leader-Follower Errands

MySQL for PCF automates orchestrating the standard lifecycle of creating, updating, and deleting leader-follower service instances.

In addition, MySQL for PCF provides three leader-follower errands. An operator can use them to control the lifecycle of a leader-follower service instance and manually intervene, when necessary, without being an expert at MySQL.

For more information, see [Leader-Follower Errands](#).

MySQL for PCF Recommended Usage and Limitations

Recommended Use Cases

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 On-Demand Plans

- Each on-demand plan instance is deployed to its own VM and is suitable for production workloads.
- The maximum-number of on-demand plan instances available to developers is set by the operator and enforced on both a global and per-plan level quota. The global quota cannot exceed 50.
- Operators can update the plan settings, including the VM size and disk size after the plans have been created. **Operators should not downsize the VMs or disk size as this can cause data loss in pre-existing instances.**
- All plans deploy a single VM of the specified size with the specified disk type.
- Cannot scale down the size of VMs on the plan once deployed (this protects against data loss).
- The default maximum number of connections is set at 750.

Resource Usage Planning for On-Demand Plans

For information on setting limits and calculating costs for on-demand service instances, see [Setting Limits for On-Demand Service Instances](#).

Availability Using Multiple AZs

MySQL for PCF v2 supports configuring multiple availability zones. At a minimum, Pivotal recommends using two availability zones, one for the broker and one for the MySQL VMs. If you plan to configure leader-follower, Pivotal recommends three availability zones as described in the [About Leader-Follower](#) page.

Downtime During Redeploys

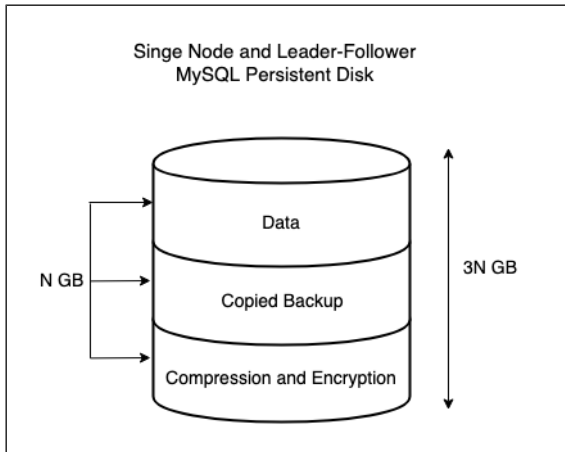
Redeploying MySQL for PCF for configuration changes or upgrades will result in MySQL being inaccessible to apps for a brief period of time.

MySQL Persistent Disk Usage

The MySQL persistent disk receives and temporarily stores and encrypts MySQL backups. This results in increased disk usage while backups are being processed.

When enabling backups for singler-node and leader-follower service plans, choose MySQL persistent disk sizes that are three times larger than the maximum amount of data an app developer using that plan will need.

The following diagram shows how persistent disk space is used.



Installing and Configuring MySQL for PCF

This topic provides instructions to operators of Pivotal Cloud Foundry (PCF) about how to install, configure, and deploy the MySQL for PCF v2.2 tile. The MySQL for PCF v2.2 service lets PCF developers create and use their own MySQL service instances on demand.

Role-Based Access in Ops Manager

Ops Manager administrators can use Role-Based Access Control (RBAC) to manage which operators can make deployment changes, view credentials, and manage user roles in Ops Manager. Therefore, your role permissions may not allow you to perform every procedure in this operator guide.

For more information about roles in Ops Manager, see [Understand Roles in Ops Manager](#).

Prepare Your Ops Manager and PCF Installation for MySQL for PCF

Before you download and install the MySQL for PCF tile, complete the following procedures:

- [Create an Application Security Group for MySQL for PCF](#)
- [Enable the Ops Manager Resurrector](#)

Create an Application Security Group for MySQL for PCF

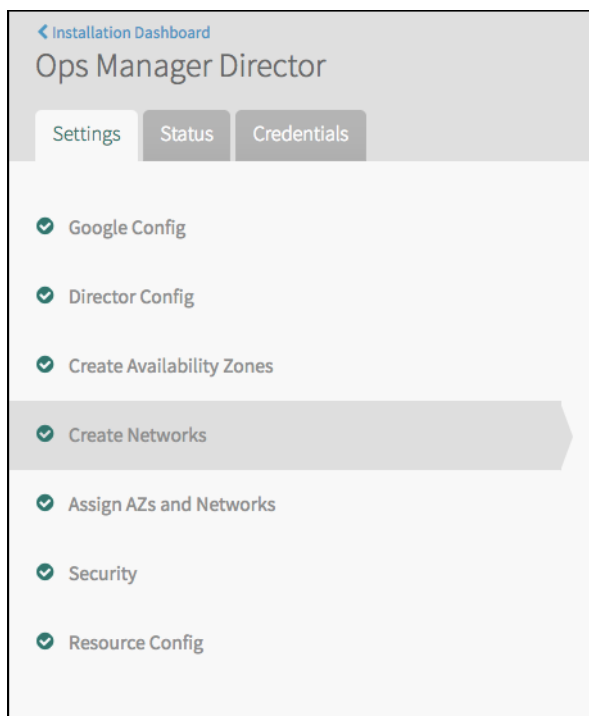
To let apps running on Cloud Foundry communicate with the MySQL service network, you must create an Application Security Group (ASG).

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

The example below follows [this procedure](#) to create an ASG.

To create an ASG for MySQL for PCF, do the following:

1. Navigate to the Ops Manager Installation Dashboard and click the **Ops Manager Director** tile.
2. Click **Create Networks**.



3. Find the network that has **Service Network** checked, and find the **CIDR** that you can use in your ASGs.

Networks

One or many IP ranges upon which your products will be deployed

▶ opsman-network

▶ ert-network

▼ services-network

Name*

services-network

☒ Service Network

Subnets

Google Network Name*

example-pcf-network/example-services-sut

CIDR*

10.0.8.0/22

Reserved IP Ranges

10.0.8.0-10.0.8.4

- Using the CIDR that you found in the above step, create a JSON file `mysql-asg.json` with the configuration below:

```
[
{
  "protocol": "tcp",
  "destination": "CIDR",
  "ports": "3306"
}
]
```

- Use the CF CLI and the JSON file that you created to create an ASG called `p.mysql`:

```
$ cf create-security-group p.mysql /mysql-asg.json
```

- Bind the ASG to the appropriate space or, to give all started apps access, bind to the `default-running` ASG set:

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

Enable the Ops Manager Resurrector

The [Ops Manager Resurrector](#) increases the availability of MySQL for PCF by restarting and resuming MySQL service in the following ways:

- Reacts to hardware failure and network disruptions by restarting VMs on active, stable hosts
- Detects operating system failures by continuously monitoring VMs and restarting them as required
- Continuously monitors the BOSH Agent running on each service instance VM and restarts the VM as required

Pivotal recommends enabling the Ops Manager Resurrector when installing MySQL for PCF. To enable the Ops Manager Resurrector, do the following:

- Navigate to the Ops Manager Installation Dashboard and click the **Ops Manager Director** tile.
- Click **Director Config**.
- Select the **Enable VM Resurrector Plugin** checkbox.
- Click **Save**.

For general information about the Ops Manager Resurrector, see [Using Ops Manager Resurrector](#).

Download and Install the Tile

1. Download the product file from [Pivotal Network](#).
2. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.
3. Under the **Import a Product** button, click + next to the version number of MySQL for PCF. This adds the tile to your staging area.
4. Click the newly-added **MySQL for PCF** tile to open its configuration panes.

Configure the Tile

Follow the steps below to configure the MySQL for PCF service. MySQL for PCF v2.2 has five service plans that deploy dedicated MySQL service instances on demand.

By default, plans 1-3 are active and plans 4 and 5 are inactive. The procedures below describe how to change these defaults.

Important: In order to re-define plans later, you must leave `broker de-registrar` checked.

Configure AZs and Networks

Follow the steps below to choose an availability zone (AZ) to run the service broker and to select networks.

1. Click **Assign AZs and Networks**.

2. Configure the fields as follows:

Field	Instructions
Place singleton jobs in	Select the AZ that you want the MySQL broker VM to run in. The broker runs as a singleton job.
Balance other jobs in	Ignore; not used.
Network	Select a subnet for the MySQL broker. This is typically the same subnet that includes the Pivotal Application Service (PAS) or Elastic Runtime component VMs. This network is represented by the Default Network in this picture .
Service Network	Select the subnet for the on-demand service instances, the Services Network in this picture . If you are adding IPsec to encrypt MySQL communication, Pivotal recommends that you deploy MySQL to its own network to avoid conflicts with services that are not IPsec compatible.



IMPORTANT: You cannot change the regions or networks after you click **Apply Changes** in the [final step](#) below.

3. Click **Save**.

Configure Service Plans

MySQL for PCF enables you to configure as many as five service plans. Each service plan has a corresponding section in the tile configuration, such as **Plan 1**, **Plan 2**, and so on.

By default, the first three plans are active and the fourth and fifth plans are inactive.

Perform the following steps for each plan that you want to use in your PCF deployment:

1. Click the section for the plan. For example, **Plan 1**.
2. Ensure that **Active** is selected.

Plan*

☐ Inactive
☒ Active

☒ Multi-node deployment

Service Plan Access*

Enable

Plan Name *

db-small

Plan Description *

This plan provides a small dedicated MySQL instance.

Plan Quota (min: 0, max: 50)

MySQL VM Type*

small (cpu: 1, ram: 2 GB, disk: 8 GB)

MySQL Persistent Disk*

5 GB

MySQL Availability Zone(s) *

☐ us-central1-a
☒ us-central1-b
☒ us-central1-c

Save

3. Configure the fields as follows:

Field	Instructions
Multi-node deployment	Check the box to configure a leader-follower plan. Leave unchecked to configure each instance as a singleton. For more information, see Configure a Leader-Follower Service Plan .
Service Plan Access	Select one of the following options: <ul style="list-style-type: none"> Enable (Default)—Gives access to all orgs, and displays the service plan to all developers in the Marketplace. Disable—Disables access to all orgs, and hides the service plan to all developers in the Marketplace. This disables creating new service instances of this plan. Manual—Lets you manually control service access with the cf CLI. For more information, see Controlling Access to Service Plans by Org.
Plan Name	Accept the default or enter a name. This is the name that appears in the Marketplace for developers.
Plan Description	Accept the default or enter a description to help developers understand plan features. Pivotal recommends adding VM type details and disk size to this field.
Plan Quota	Enter the maximum number of service instances that can exist at one time.
MySQL VM Type	Select a VM type. The plan creates service instances of this size.
	Select a disk size. This disk stores the MySQL messages.

MySQL Persistent Disk	<p>Note: If you intend to enable backups, choose a persistent disk type that is three times larger than the maximum amount of data that developers will store on the disk. For more information, see MySQL Persistent Disk Usage.</p>
MySQL Availability Zone(s)	<p>If you check the Multi-node deployment box above to enable a leader-follower plan, choose one of the following:</p> <ul style="list-style-type: none"> To leverage the full HA advantages of leader-follower, select two AZs. You must choose AZs different from the one the broker is in. To leverage some HA advantages, select one AZ. Using only one AZ for a Multi-node deployment greatly reduces the ability to recover from an outage and benefit from such a deployment. <p>If you do not check Multi-node deployment, select one AZ.</p> <p>Note: To prevent orphaning a disk, only migrate service instances to a different plan if the new plan includes all of the AZs in the existing plan.</p>

4. Click **Save**.

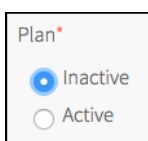
Warning: If you expect your developers to upgrade from one plan to another, do not place the plans in separate AZs. For example, if you create Plan 1 in AZ1 and Plan 2 in AZ2, developers receive an error and cannot continue if they try to upgrade from Plan 1 to Plan 2. This prevents them from losing their data by orphaning their disk in AZ1.

To learn how to manually migrate the data from one AZ to another, contact [Pivotal Support](#).

(Optional) Deactivate Service Plan

Follow the steps below to deactivate a service plan:

- If the service plan has existing service instances, perform the following steps:
 - Click the section for the plan. For example, **Plan 2**.
 - Under **Service Plan Access**, select **Disable**.
 - Click **Save**.
 - Return to the Ops Manager Installation Dashboard and click **Apply Changes** to redeploy.
 - When the PCF deployment has redeployed, use the cf CLI or Apps Manager to delete all existing service instances on the service plan.
 - Return to the MySQL for PCF tile configuration.
- Click the section for the plan. For example, **Plan 2**.
- Click **Inactive**.



4. Click **Save**.

Configure Global Settings

Follow the steps below to determine if service instances are assigned public IP addresses and to set the total number of service instances allowed across all plans.

- Click **Settings**.

Provide public IP addresses to all Service VMs

☐

Allow outbound internet access from service instances (IaaS-dependent)

Maximum service instances (min: 0, max: 50) *

3

Save

2. Configure the fields as follows:

Field	Instructions
Provide public IP addresses to all Service VMs	Select this checkbox: <ul style="list-style-type: none"> If the service instances need an external backup, blobstore, or syslog storage If you have configured BOSH to use an external blobstore.
Maximum service instances	Enter the global quota for all on-demand instances summed across every on-demand plan. For information about determining global quotas, see Service Plan Recommended Usage and Limitations .

3. Click **Save**.

Configure MySQL

Follow the steps below to set MySQL defaults and enable developers to customize their instances:

1. Click **MySQL Configuration**

☐ Enable Lower Case Table Names

☐ Allow Developers To Override Lower Case Table Names

☐ Enable Local Infile


Save

2. Configure the fields as follows:

Field	Instructions
Enable Lower Case Table Names	Select this checkbox to store all table names in lowercase. Selecting this sets the MySQL server system variable <code>lower_case_table_names</code> to 1 on all MySQL for PCF instances by default. To allow developers to override this default, see the checkbox below. For more information, see lower_case_table_names in the MySQL documentation.
Allow Developers To Override Lower Case Table Names	Select this checkbox to allow developers to override the default Enable Lower Case Table Names value set above, when they are creating a new service instance. For more information, see Optional Parameters for the MySQL for PCF Service Instances .
Enable Local Infile	Select this checkbox to enable data downloading from the client's local file system. For more information about local data loading capability, see Security Issues with LOAD DATA LOCAL in the MySQL 5.7 Reference Manual.

Configure Backups

Follow the steps below to configure automated backups:

 **Note:** As of v2.2.0, the MySQL for PCF tile no longer includes the option to disable automated backups. You must configure automated backups.

1. Click **Backups**.

Configure blob store for MySQL backups

Backup configuration*

☐ Ceph or Amazon S3
☐ SCP
☐ GCS
☐ Azure

Save

2. Consult [About Automated Backups](#) in *Backing Up and Restoring* to learn about how automated backups work.
3. Select a **Backup configuration** and perform the steps in the appropriate section of *Backing Up and Restoring*:
 - [Option 1: Back up with SCP](#)—MySQL for PCF runs an SCP command that secure-copies backups to a VM or physical machine operating outside of PCF. SCP stands for secure copy protocol, and offers a way to securely transfer files between two hosts. The operator provisions the backup machine separately from their PCF installation. This is the fastest option.
 - [Option 2: Back up to Ceph or S3](#)—MySQL for PCF runs an [Amazon S3](#) client that saves backups to an S3 bucket, a [Ceph](#) storage cluster, or another S3-compatible endpoint certified by Pivotal.
 - [Option 3: Back up to GCS](#)—MySQL for PCF runs an [GCS](#) SDK that saves backups to an Google Cloud Storage bucket.
 - [Option 4: Back up to Azure Storage](#)—MySQL for PCF runs an [Azure](#) SDK that saves backups to an Azure storage account.

Configure Monitoring

Follow the steps below to enable different types of monitoring and logging available in the MySQL service.

1. Click **Monitoring**.

Metrics Polling Interval (min: 10) *

30

☐ Enable User Statistics Logging
☐ Enable Server Activity Logging
☐ Enable Read Only Admin User

Save

2. Configure the fields as follows:

Field	Instructions
Metrics Polling Interval	Set this time, in seconds, to determine the frequency that the monitor polls for metrics. All service instances emit metrics about the health and status of the MySQL server.
Enable User Statistics Logging	Select this checkbox to better understand server activity and identify sources of load on a MySQL server. For more information about user statistics, see User Statistics Documentation .
Enable Server Activity Logging	Select this checkbox to record who connects to the servers and what queries are processed using the Percona Audit Log Plugin. For more information, see the Percona Documentation .
Enable Read Only Admin User	<p>Select this checkbox to create a read-only admin user, <code>roadmin</code>, on each service instance. This user can be used for auditing and monitoring without risking mutating or changing any data, because the <code>roadmin</code> user cannot make changes.</p> <p>To retrieve the credentials for the <code>roadmin</code> user, see Retrieve Admin and Read-Only Admin Credentials for a Service Instance.</p>

The read-only admin user is always `roadmin`, but the password varies by service instance.

3. Click **Save**.

Configure System Logging

Follow the steps below to enable system logging for the MySQL broker and service instance VMs. Logs use RFC5424 format.

1. Click **Syslog**.
2. Click **Yes**.

Do you want to configure Syslog for the MySQL service?

☐ No
☒ Yes

Address

Port

Transport Protocol

TCP

☒ Enable TLS

Permitted Peer

SSL Certificate

3. Configure the fields as follows:

Field	Instructions
Address	Enter the address or host of the syslog server for sending logs, for example, <code>logmanager.example.com</code> .
Port	Enter the port of the syslog server for sending logs, for example, <code>29279</code> .
Transport Protocol	Select the protocol over which you want system logs. Pivotal recommends using TCP.
Enable TLS	If you select TCP, you can also select to send logs encrypted over TLS.
Permitted Peer	Enter either the accepted fingerprint, in SHA1, or the name of the remote peer, for example, <code>*.example.com</code>
SSL Certificate	Enter the SSL Certificate(s) for the syslog server. This ensures the logs are transported securely.

IMPORTANT: If your syslog server is external to PCF, you might need to select **Provide public IP addresses to all Service VMs** on the **Settings** page.

4. Click **Save**.

Verify Stemcell Version and Apply All Changes

1. Click **Stemcell**.


2. Verify and, if necessary, import a new stemcell version.

For more information, see about importing the stemcell for your IaaS: [AWS](#), [Azure](#), [GCP](#), or [vSphere](#).

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

Backing Up and Restoring On-Demand MySQL for PCF

This topic describes how to configure automated backups for MySQL for Pivotal Cloud Foundry (PCF), and how to manually restore a MySQL service instance from a backup.

 **Note:** As of v2.2.0, the MySQL for PCF tile no longer includes the option to disable automated backups. You must configure automated backups.

About Automated Backups

Automated backups do the following:

- Periodically create and upload backups suitable for restoring all of the databases used by the service instance.
- Operate without locking apps out of the database; no downtime.
- Include a metadata file that contains the critical details of the backup, including the calendar time of the backup.
- Encrypt backups within the MySQL for PCF VM; unencrypted data is never transported outside the MySQL for PCF deployment.

Backup Files and Metadata

When MySQL for PCF runs a backup, it uploads two files with Unix epoch-timestamped filenames of the form `mysql-backup-TIMESTAMP`:


- The encrypted data backup file `mysql-backup-TIMESTAMP.tar.gpg`
- A metadata file `mysql-backup-TIMESTAMP.txt`

The metadata file contains information about the backup and looks like this:

```
ibbackup_version = 2.4.5
end_time = 2017-04-24 21:00:03
lock_time = 0
binlog_pos =
incremental = N
encrypted = N
server_version = 5.7.16-10-log
start_time = 2017-04-24 21:00:00
tool_command = --user=admin --password=... --stream=tar tmp/
innodb_from_lsn = 0
innodb_to_lsn = 2491844
format = tar
compact = N
name =
tool_name = innobackupex
partial = N
compressed = N
uuid = fd13cf26-2930-11e7-871e-42010a000807
tool_version = 2.4.5
```

Within this file, the most important items are the `start_time` and the `server_version` entries.

The backup process does not interrupt MySQL service, but backups only reflect transactions that completed before their `start_time`.

 **Note:** Although both `compressed` and `encrypted` show as `N` in this file, the backup uploaded by MySQL for PCF is both compressed and encrypted. This is a known issue.

About Configuring Automated Backups

MySQL for PCF automatically backs up databases to external storage.

- **How and where**—There are four options for how automated backups transfer backup data and where the data saves to:
 - **Option 1: Back up with SCP**—MySQL for PCF runs an SCP command that secure-copies backups to a VM or physical machine operating outside of PCF. SCP stands for secure copy protocol, and offers a way to securely transfer files between two hosts. The operator provisions the backup machine

separately from their PCF installation. This is the fastest option.

- [Option 2: Back up to Ceph or S3](#)—MySQL for PCF runs an [Amazon S3](#) client that saves backups to an S3 bucket, a [Ceph](#) storage cluster, or another S3-compatible endpoint certified by Pivotal.
 - [Option 3: Back up to GCS](#)—MySQL for PCF runs an [GCS](#) SDK that saves backups to an Google Cloud Storage bucket.
 - [Option 4: Back up to Azure Storage](#)—MySQL for PCF runs an [Azure](#) SDK that saves backups to an Azure storage account.
- **When**—Backups follow a schedule that you specify with a [cron](#) expression.
 - **What is backed up**—Each MySQL instance backs up its entire MySQL data directory, consistent to a specific point in time.

To configure automated backups, follow the procedures below according to the option you choose for external storage.

Option 1: Back Up with SCP

SCP enables the operator to use any desired storage solution on the destination VM.

To back up your database using SCP, complete the following procedures:

- [Create a Public and Private Key Pair](#)
- [Configure Backups in Ops Manager](#)

Create a Public and Private Key Pair

MySQL for PCF accesses a remote host as a user with a private key for authentication. Pivotal recommends that this user and key pair be solely for MySQL for PCF.

1. Determine the remote host that you will be using to store backups for MySQL for PCF. Ensure that the MySQL service instances can access the remote host.



Note: Pivotal recommends using a VM outside the PCF deployment for the destination of SCP backups. As a result you might need to enable public IPs for the MySQL VMs.

2. (Recommended) Create a new user for MySQL for PCF on the destination VM.
3. (Recommended) Create a new public and private key pair for authenticating as the above user on the destination VM.

Configure Backups in Ops Manager

Use Ops Manager to configure MySQL for PCF to backup using SCP.

1. In Ops Manager, open the MySQL for PCF tile **Backups** pane.
2. Select **SCP**.

Backup configuration*

☐ Ceph or Amazon S3
☒ SCP

Username*

Private Key*

Hostname*

Destination Directory*

SCP Port*

Cron Schedule*

Fingerprint

☐ Enable Email Alerts

3. Fill in the fields as follows:

- **Username**—Enter the user that you created above.
- **Private Key**—Enter the private key that you created above. The public key should be stored for ssh and scp access on the destination VM.
- **Hostname**—Enter the IP or DNS entry that should be used to access the destination VM.
- **Destination Directory**—Enter the directory in which MySQL for PCF should upload backups.
- **Cron Schedule**—Enter a `cron` schedule, using standard [cron syntax](#), to take backups of each service instance.
- **Fingerprint**—Enter the fingerprint of the destination VM's public key. This helps to detect any changes to the destination VM.
- **Enable Email Alerts**—Select to receive email notifications when a backup failure occurs. Also verify that you have done the following:
 - Added all users who need to be notified about failed backups to System org and System space
 - Configured Email Notifications in Pivotal Application Service (PAS) or Elastic Runtime, see *Configure Email Notifications* for your IaaS: [AWS](#), [Azure](#), [GCP](#), [OpenStack](#), or [vSphere](#).

Option 2: Back Up to Ceph or S3

To back up your database on Ceph or Amazon S3, complete the following procedures:

- [Create a Policy and Access Key](#)
- [Configure Backups in Ops Manager](#)

Create a Policy and Access Key

MySQL for PCF accesses your S3 store through a user account. Pivotal recommends that this account be solely for MySQL for PCF. You must apply a minimal policy that lets the user account upload backups to your S3 store.

1. Create a policy for your MySQL for PCF user account.

Policy creation varies depending on the S3 provider. Follow the relevant procedure to give MySQL for PCF the following permissions:

- List and upload to buckets
- (Optional) Create buckets in order to use buckets that do not already exist

For example, in AWS, to create a new custom policy, go to **IAM > Policies > Create Policy > Create Your Own Policy** and paste in the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServiceBackupPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:CreateBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::MY_BUCKET_NAME/*",
        "arn:aws:s3:::MY_BUCKET_NAME"
      ]
    }
  ]
}
```

2. (Recommended) Create a new user for MySQL for PCF and record its Access Key ID and Secret Access Key, the user credentials.
3. Attach the policy you created to the AWS user account that MySQL for PCF will use to access S3 (**IAM > Policies > Policy Actions > Attach**).

Configure Backups in Ops Manager

Use Ops Manager to connect MySQL for PCF to your S3 account.

1. In Ops Manager, open the MySQL for PCF tile **Backups** pane.
2. Select **Ceph** or **Amazon S3**.

☒ Ceph or Amazon S3

Access Key ID *

Secret Access Key *

Endpoint URL

Region

Bucket Name *

Bucket Path *

Cron Schedule *

☐ Enable Email Alerts

3. Fill in the fields as follows:

- **Access Key ID** and **Secret Access Key**—Enter the S3 Access Key ID and Secret Access Key from above.
- **Endpoint URL**—Enter the S3 compatible endpoint URL for uploading backups. URL must start with `http://` or `https://`. The default is <https://s3.amazonaws.com>
- **Region**—Enter the region where your bucket is located or the region where you want a bucket to be created. If the bucket does not already exist, it is created automatically.
- **Bucket Name**—Enter the name of your bucket. Do not include an `s3://` prefix, a trailing `/`, or underscores. Pivotal recommends using the naming convention `DEPLOYMENT-backups`, such as `sandbox-backups`.
- **Bucket Path**—Enter the path in the bucket to store backups. Do not include a trailing `/`. Pivotal recommends using `mysql-v2`.
- **Cron Schedule**—Enter a `cron` schedule, using standard [cron syntax](#), to take backups of each service instance.
- **Enable Email Alerts**—Select to receive email notifications when a backup failure occurs. Also verify that you done the following:
 - Added all users who need to be notified about failed backups to System org and System space
 - Configured Email Notifications in PAS (or Elastic Runtime), see *Configure Email Notifications* for your IaaS: [AWS](#), [Azure](#), [GCP](#), [OpenStack](#), or [vSphere](#).

Option 3: Back Up to GCS

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

- [Create a Policy and Access Key](#)
- [Configure Backups in Ops Manager](#)

Create a Policy and Access Key

MySQL for PCF accesses your GCS store through a service account. Pivotal recommends that this account be solely for MySQL for PCF. You must apply a minimal policy that lets the user account upload backups to your GCS store.

1. In the GCS console, create a new service account for MySQL for PCF: **IAM and Admin > Service Accounts > Create Service Account**.

MySQL for PCF needs the following permissions for this account:

- List and upload to buckets
- (Optional) Create buckets in order to use buckets that do not already exist

2. Enter a unique name in the **Service account name** field, such as `MySQL-for-PCF`.
3. In the **Roles** dropdown, grant the `MySQL-for-PCF` service account the **Storage Admin** role.
4. Check the **Furnish a new private key** box so that a new key is created and downloaded.
5. Click **Create** and take note of the name and location of the service account JSON file that is downloaded.

Configure Backups in Ops Manager

Use Ops Manager to connect MySQL for PCF to your GCS account.

1. In Ops Manager, open the MySQL for PCF tile **Backups** pane.
2. Select **GCS**.

3. Fill in the fields as follows:
 - **Project ID**—Enter the Project ID for the Google Cloud project that you are using.
 - **Bucket Name**—Enter the bucket name in which to upload.
 - **Service Account JSON**—Enter the contents of the service account json file that you downloaded when creating a service account above.
 - **Cron Schedule**—Enter a `cron` schedule, using standard [cron syntax](#), to take backups of each service instance.
 - **Enable Email Alerts**—Select to receive email notifications when a backup failure occurs. Also verify that you done the following:
 - Added all users who need to be notified about failed backups to System org and System space
 - Configured Email Notifications in PAS (or Elastic Runtime), see *Configure Email Notifications* for your IaaS: [AWS](#), [Azure](#), [GCP](#), [OpenStack](#), or [vSphere](#).

Option 4: Back Up to Azure Storage

Complete the following steps to back up your database to your Azure Storage account.

1. In Ops Manager, open the MySQL for PCF tile **Backups** pane.

Backup configuration *

☐ Ceph or Amazon S3

☐ SCP

☐ GCS

☒ Azure

Account *

Azure Storage Access Key *

Container Name *

Destination Directory *

Blob Store Base URL

Cron Schedule *

0 */8 * * *

☐ Enable Email Alerts

2. Select **Azure**.

3. Fill in the fields as follows:

- **Account**—Enter the Azure Storage account name that you are using.
- **Azure Storage Access Key**—Enter one of the storage access keys that can be used to access the Azure Storage account.
- **Container Name**—Enter the container name that backups should be uploaded to.
- **Destination Directory**—Enter the directory in which backups should be uploaded to inside of the Container.
- **Blob Store Base URL**—By default, backups are sent to the public Azure blob store. To use an on-premise blob store, enter the hostname of the blob store.
- **Cron Schedule**—Enter a `cron` schedule, using standard [cron syntax](#), to take backups of each service instance.
- **Enable Email Alerts**—Select to receive email notifications when a backup failure occurs. Also verify that you done the following:
 - Added all users who need to be notified about failed backups to System org and System space
 - Configured Email Notifications in PAS (or Elastic Runtime), see *Configure Email Notifications* for your IaaS: [AWS](#), [Azure](#), [GCP](#), [OpenStack](#), or [vSphere](#).

Manual Backup

MySQL for PCF v2.2 disables remote admin access to MySQL databases. To access your MySQL database to perform a manual backup, you must create a service key for each service instance you want to back up.

This backup acquires a global read lock on all tables, but does not hold it for the entire duration of the dump.

Perform the following steps to back up your MySQL for PCF data manually:

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 -c '{"read-only":true}'`

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 \
-c '{"read-only":true}'
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 --username=USERNAME --password=PASSWORD \
--host=MYSQL-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-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 --username=abcdefghijkm --password=123456789 \
--host=10.10.10.8 \
--all-databases \
--single-transaction > spring-db-backup.sql
```

Restore a Service Instance from Backup

Restoring MySQL for PCF from backup is a manual process primarily intended for disaster recovery. Restoring a MySQL for PCF service instance replaces all of its data and running state.

To restore a MySQL for PCF instance from an offsite backup, download the backup and restore to a new instance by following these procedures:

- [Download the Backup Artifact](#)
- [Restore the Service Instance](#)

Download the Backup Artifact

Download the backup artifact from your blob storage.

These instructions assume that you are using AWS S3 as your backup destination. If you are using a different backup destination, steps 5, 6, and 8 are different. See the documentation for your backup provider for the correct procedure.

Perform the following steps to download the backup artifact from an AWS S3 bucket:

These instructions assume that you are using S3 as your backup destination. If you are using one of the other backup options, steps 5, 6, and 8 are different.

1. Run `cf service` to determine the GUID associated with the service instance that you want to restore:

```
$ cf service MY-INSTANCE-NAME --guid
12345678-90ab-cdef-1234-567890abcdef
```

2. Follow the [Advanced Troubleshooting](#) instructions to log in to the Ops Manager VM, where you can run the BOSH CLI v2.
3. From the Ops Manager VM, download the manifest for the service instance deployment by specifying the deployment name as `service-instance_GUID` and a filename for the manifest. For example:

```
$ bosh2 -e my-env -d service-instance_12345678-90ab-cdef-1234-567890abcdef manifest > ./manifest.yml
```

4. Inspect the downloaded manifest and record the following properties for later:
 - `properties.cf-mysql-backup.symmetric_key` also referred to as an encryption key - this will be used later to decrypt the backup artifact.
 - `properties.service-backup.destinations[0].config.bucket_name` which is the bucket that the backups are uploaded to.
 - `properties.service-backup.destinations[0].config.bucket_path` the path within the bucket above.
5. Log in to the AWS CLI. For information and to download the CLI, see [AWS Command Line Interface](#).
6. List the available backups for the instance, replacing `BUCKET-NAME` and `BUCKET-PATH` with values from above. The artifacts are sorted by time.

```
$ aws s3 ls --recursive s3://BUCKET-NAME/BUCKET-PATH/service-instance_12345678-90ab-cdef-1234-567890abcdef/
```

7. Choose the most recent backup file, or an older backup you want to restore from. The backups are timestamped in the filename and have a `.gpg` extension.
8. Download the selected backup:

```
$ aws s3 cp s3://BUCKET-NAME/BUCKET-PATH/service-instance_12345678-90ab-cdef-1234-567890abcdef/YEAR/MONTH/DATE/mysql-backup-1489168980-0.tar.gpg ./a-local-pa
```



Note: You can also log in to AWS and download S3 backups from a browser.

Restore the Service Instance

Because restoring a service instance is destructive, Pivotal recommends that you restore to a new and unused service instance.

These instructions assume you have a backup downloaded.

To restore the backup to a new service instance, follow these procedures:

- [Create and Prepare a New Service Instance for Restore](#)
- [Run the Restore Utility](#)

Create and Prepare a New Service Instance for Restore

1. Determine the service plan associated with the old service instance:

```
$ cf service OLD-INSTANCE-NAME
```

2. [Create a new service instance](#) using the same service plan to restore to:

```
$ cf create-service p.mysql db-small NEW-SERVICE-INSTANCE
```

The persistent disk size of this service plan must be three times larger than the maximum size than the disk you intend for developers. For more information, see [MySQL Persistent Disk Usage](#).

3. Monitor the status of the service instance creation; it may take several minutes as the VM is created.

```
$ cf service NEW-SERVICE-INSTANCE
```

4. Determine the GUID associated with your service instance:

```
$ cf service NEW-SERVICE-INSTANCE --guid
```

5. From the Ops Manager VM, download the manifest for the service instance deployment by specifying the deployment name as `service-instance_GUID` and a filename for the manifest. For example:

```
$ bosh2 -e my-env -d service-instance_12345678-90ab-cdef-1234-567890abcdef manifest > ./manifest.yml
```

6. Determine and record the MySQL admin password:

```
$ cat ./manifest | grep admin_password
```

7. Find and record the MySQL instance name and GUID from BOSH:

```
$ bosh2 -e my-env -d my-dep instances
```

8. Copy the downloaded backup to the new service instance:

```
$ bosh2 -e my-env -d my-dep scp mysql-backup-TIMESTAMP.tar.gpg BOSH-INSTANCE:DESTINATION-PATH
```

- BOSH-INSTANCE is `mysql/INSTANCE-GUID`. For example, `mysql/d7ff8d46-c3e8-449f-aea7-5a05b0a1929c`.
- DESTINATION-PATH is where the backup file saves on the BOSH VM. For example, `/tmp/`.

9. Use the BOSH CLI v2 to SSH in to the newly created MySQL service instance. For more information, see [BOSH SSH](#).

10. After securely logging in to MySQL, run `sudo su` to become root:

```
$ sudo su
```

Run the Restore Utility

⚠ warning: This is a destructive action and should only be run on a new and unused service instance.

1. Run the restore utility to restore the backup artifact into the data directory. This process:

- deletes any existing data
- decrypts and untars the backup artifact
- restores the backup artifact into the mysql data directory

```
$ mysql-restore --encryption-key ENCRYPTION-KEY --mysql-username admin --mysql-password ADMIN-PASSWORD --restore-file RESTORE-FILE-PATH
```

where:

- `ENCRYPTION-KEY` is the `symmetric-key` value from above
- `ADMIN-PASSWORD` is the `admin_password` value from above
- `RESTORE-FILE-PATH` is the full path on the MySQL VM where the backup artifact exists

2. Exit the MySQL VM.

3. Determine if the app is currently bound to a MySQL service instance:

```
$ cf services
```

4. If the previous step shows that the app is currently bound to a MySQL instance, unbind it:

```
$ cf unbind-service MY-APP OLD-SERVICE-INSTANCE
```

5. Update your CF app to bind to the new service instance:

```
$ cf bind-service MY-APP NEW-SERVICE-INSTANCE
```

6. Restage your CF app to make the changes take effect:

```
$ cf restage MY-APP
```

Your app should be running and able to access the restored data.

Setting Limits for On-Demand Service Instances

On-demand provisioning is intended to accelerate app development by eliminating the need for development teams to request and wait for operators to create a service instance. However, to control costs, operations teams and administrators must ensure responsible use of resources.

There are several ways to control the provisioning of on-demand service instances by setting various **quotas** at these levels:

- [Global](#)
- [Plan](#)
- [Org](#)
- [Space](#)

After you set quotas, you can:

- [View Current Org and Space-level Quotas](#)
- [Monitor Quota Use and Service Instance Count](#)
- [Calculate Resource Costs for On-Demand Plans](#)

Create Global-level Quotas

Each Pivotal Cloud Foundry (PCF) service has a separate service broker. A global quota at the service level sets the maximum number of service instances that can be created by a given service broker. If a service has more than one plan, then the number of service instances for all plans combined cannot exceed the global quota for the service.

The operator sets a global quota for each PCF service independently. For example, if you have Redis for PCF and RabbitMQ for PCF, you must set a separate global service quota for each of them.

When the global quota is reached for a service, no more instances of that service can be created unless the quota is increased, or some instances of that service are deleted.

Create Plan-level Quotas

A service may offer one or more plans. You can set a separate quota per plan so that instances of that plan cannot exceed the plan quota. For a service with multiple plans, the total number of instances created for all plans combined cannot exceed the [global quota](#) for the service. If the plan quota field is blank, the plan quota is set to the global quota by default.

When the plan quota is reached, no more instances of that plan can be created unless the plan quota is increased or some instances of that plan are deleted.

Create and Set Org-level Quotas

An org-level quota applies to all PCF services and sets the maximum number of service instances an organization can create within PCF. For example, if you set your org-level quota to 100, developers can create up to 100 service instances in that org using any combination of PCF services.

When this quota is met, no more service instances of any kind can be created in the org unless the quota is increased or some service instances are deleted.

To create and set an org-level quota, do the following:

1. Run this command to create a quota for service instances at the org level:

```
cf create-quota QUOTA-NAME -m TOTAL-MEMORY -i INSTANCE-MEMORY -r ROUTES -s SERVICE-INSTANCES --allow-paid-service-plans
```

Where:

- `QUOTA-NAME` —A name for this quota
- `TOTAL-MEMORY` —Maximum memory used by all service instances combined
- `INSTANCE-MEMORY` —Maximum memory used by any single service instance
- `ROUTES` —Maximum number of routes allowed for all service instances combined

- `SERVICE-INSTANCES` —Maximum number of service instances allowed for the org

For example:

```
cf create-quota myquota -m 1024mb -i 16gb -r 30 -s 50 --allow-paid-service-plans
```

2. Associate the quota you created above with a specific org by running the following command:

```
cf set-quota ORG-NAME QUOTA-NAME
```

For example:

```
cf set-quota dev_org myquota
```

For more information on managing org-level quotas, see [Creating and Modifying Quota Plans](#).

Create and Set Space-level Quotas

A space-level service quota applies to all PCF services and sets the maximum number of service instances that can be created within a given space in PCF. For example, if you set your space-level quota to 100, developers can create up to 100 service instances in that space using any combination of PCF services.

When this quota is met, no more service instances of any kind can be created in the space unless the quota is updated or some service instances are deleted.

To create and set a space-level quota, do the following:

1. Run the following command to create the quota:

```
cf create-space-quota QUOTA-NAME -m TOTAL-MEMORY -i INSTANCE-MEMORY -r ROUTES -s SERVICE-INSTANCES --allow-paid-service-plans
```

Where:

- `QUOTA-NAME` —A name for this quota
- `TOTAL-MEMORY` —Maximum memory used by all service instances combined
- `INSTANCE-MEMORY` —Maximum memory used by any single service instance
- `ROUTES` —Maximum number of routes allowed for all service instances combined
- `SERVICE-INSTANCES` —Maximum number of service instances allowed for the org

For example:

```
cf create-space-quota myspacequota -m 1024mb -i 16gb -r 30 -s 50 --allow-paid-service-plans
```

2. Associate the quota you created above with a specific space by running the following command:

```
cf set-space-quota SPACE-NAME QUOTA-NAME
```

For example:

```
cf set-space-quota myspace myspacequota
```

For more information on managing space-level quotas, see [Creating and Modifying Quota Plans](#).

View Current Org and Space-level Quotas

To view **org** quotas, run the following command.

```
cf org ORG-NAME
```

To view **space** quotas, run the following command:

cf space SPACE-NAME

For more information on managing org and space-level quotas, see the [Creating and Modifying Quota Plans](#).

Monitor Quota Use and Service Instance Count

Service-level and plan-level quota use, and total number of service instances, are available through the on-demand broker metrics emitted to Loggregator. These metrics are listed below:

Metric Name	Description
<code>on-demand-broker/SERVICE-NAME/quota_remaining</code>	Quota remaining for all instances across all plans
<code>on-demand-broker/SERVICE-NAME/PLAN-NAME/quota_remaining</code>	Quota remaining for a specific plan
<code>on-demand-broker/SERVICE-NAME/total_instances</code>	Total instances created across all plans
<code>on-demand-broker/SERVICE-NAME/PLAN-NAME/total_instances</code>	Total instances created for a specific plan

Note: Quota metrics are not emitted if no quota has been set.

You can also view service instance usage information in Apps Manager. For more information, see [Reporting Instance Usage with Apps Manager](#).

Calculate Resource Costs for On-Demand Plans

On-demand plans use dedicated VMs, disks, and various other resources from an IaaS, such as AWS. To calculate maximum resource cost for plans individually or combined, you multiply the quota by the cost of the resources selected in the plan configuration(s). The specific costs depend on your IaaS.

To view configurations for your MySQL for PCF on-demand plan, do the following:

1. Navigate to **Ops Manager Installation Dashboard > MySQL for Pivotal Cloud Foundry v2 > Settings**.
2. Click the section for the plan you want to view. For example, **Plan 1**.

The image below shows an example that includes the VM type and persistent disk selected for the server VMs, as well as the quota for this plan.

Plan Quota (min: 0, max: 50)

MySQL VM Type*

MySQL Persistent Disk*

Note: Although operators can limit on-demand instances with plan quotas and a global quota, as described in the above topics, IaaS resource usage still varies based on the number of on-demand instances provisioned.

Calculate Maximum Resource Cost Per On-Demand Plan

To calculate the maximum cost of VMs and persistent disk for each plan, do the following calculation:

plan quota x cost of selected resources

For example, if you selected the options in the above image, you have selected a VM type **micro** and a persistent disk type **20 GB**, and the plan quota is **15**. The VM and persistent disk types have an associated cost for the IaaS you are using. Therefore, to calculate the maximum cost of resources for this plan, multiply the cost of the resources selected by the plan quota:

$(15 \times \text{cost of micro VM type}) + (15 \times \text{cost of 20 GB persistent disk}) = \text{max cost per plan}$

Calculate Maximum Resource Cost for All On-Demand Plans

To calculate the maximum cost for all plans combined, add together the maximum costs for each plan. This assumes that the sum of your individual plan quotas is less than the global quota.

Here is an example:

$(\text{plan1 quota} \times \text{plan1 resource cost}) + (\text{plan2 quota} \times \text{plan2 resource cost}) = \text{max cost for all plans}$

Calculate Actual Resource Cost of all On-Demand Plans

To calculate the current actual resource cost across all your on-demand plans:

1. Find the number of instances currently provisioned for each active plan by looking at the `total_instance` [metric](#) for that plan.
2. Multiply the `total_instance` count for each plan by that plan's resource costs. Record the costs for each plan.
3. Add up the costs noted in Step 2 to get your total current resource costs.

For example:

$(\text{plan1 total_instances} \times \text{plan1 resource cost}) + (\text{plan2 total_instances} \times \text{plan2 resource cost}) = \text{current cost for all plans}$

Controlling Access to Service Plans by Org

This topic gives instructions for controlling access to service plans by org. You can also set limits for the number of service instances globally and per plan. For more information, see [Setting Limits for On-Demand Instances](#).

Control Access to Service Plan by Orgs

MySQL for PCF v2 allows you to control which CF orgs are able to access specific service plans. By default, active service plans are visible to all orgs. Controlling which orgs have access to a specific service plan allows you to ensure that the resource-intensive service plans are only available to the orgs that explicitly need them.

To configure MySQL for PCF v2 to control service-plan access, do the following:

1. Set the **Service Plan Access** field to **Manual** on any active service plan.
For more information, see [Configure Active Service Plans](#).
2. Click **Save**.
3. Return to the Ops Manager Installation Dashboard and click **Apply Changes**.
4. For each org that you want to use the service plan, do the following:
 - a. Log in to the Cloud Foundry Command Line Interface (cf CLI) as an admin user:
`cf login`
 - b. Enable service access to the org:
`cf enable-service-access p.mysql -p PLAN -o ORGANIZATION`

Where:

`PLAN`: The name of the specific plan to enable, set to manual in step 1 above.

`ORGANIZATION`: The name of the org that needs access to `PLAN`

For example,

```
$ cf enable-service-access p.mysql -o prodteam -p db-large
Enabling access to plan db-large of service p.mysql for org prodteam as admin...
OK
```

The org is now able to use the plan.

For information on modifying and viewing service-plan access, see [Access Control](#).

Monitoring and KPIs for On-Demand MySQL for PCF

This topic explains how to monitor the health of the MySQL for Pivotal Cloud Foundry (PCF) service using the logs, metrics, and Key Performance Indicators (KPIs) generated by MySQL for PCF component VMs.

For general information about logging and metrics in PCF, see [Logging and Metrics](#).

About Metrics

Metrics are regularly-generated log messages that report measured component states. The metrics polling interval is 30 seconds for MySQL instances and 60 seconds for the service broker.

Metrics are long, single lines of text that follow the format:

```
origin:"p.mysql" eventType:ValueMetric timestamp:1496776477930669688 deployment:"service-instance_2b5a001f-2bf3-460c-ae6-fd2253f9fb0c" job:"mysql" index:"b09df494-b731-4d06-a4b0-c2985ceedf4c"
```

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.

KPIs for MySQL Service Instances

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

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

For a list of all MySQL for PCF component metrics, see [All MySQL Metrics](#).

Server Availability

/p.mysql/available	
Description	<p>MySQL Server is currently responding to requests, which indicates if the component is available.</p> <p>Use: If the server does not emit heartbeats, it is offline.</p> <p>Origin: Doppler/Firehose</p> <p>Type: boolean</p> <p>Frequency: 30 s</p>
Recommended measurement	Average over last 5 minutes
Recommended alert thresholds	<p>Yellow warning: N/A</p> <p>Red critical: < 1</p>
Recommended response	Check the MySQL Server logs for errors. You can find the instance by targeting your MySQL deployment with BOSH and inspecting logs for the instance. For more information, see Failing Jobs and Unhealthy Instances .

Persistent and Ephemeral Disk Used

/p.mysql/system/persistent_disk_used_percent and /p.mysql/system/ephemeral_disk_used_percent	
Description	<p>The percentage of disk used on the persistent and ephemeral file systems.</p> <p>Use: MySQL cannot function correctly if there isn't sufficient free space on the file systems. Use these metrics to ensure that you have disks large enough for your user base.</p> <p>Origin: Doppler/Firehose Type: Percent Frequency: 30 s (default)</p>
Recommended measurement	<p>Max of persistent disk used of all of nodes and Maximum ephemeral disk used of all nodes</p>
Recommended alert thresholds	<p>Yellow warning: > 40%</p> <p>Red critical: > 45%</p>
Recommended response	Upgrade the service instance to a plan with larger disk capacity.

Connections

/p.mysql/net/max_connections /p.mysql/net/max_used_connections	
Description	<p>The maximum number of connections used over the maximum permitted number of simultaneous client connections.</p> <p>Use: If the number of connections drastically changes or if apps are unable to connect, there might be a network or app issue.</p> <p>Origin: Doppler/Firehose Type: count Frequency: 30 s</p>
Recommended measurement	<code>max_used_connections</code> / <code>max_connections</code>
Recommended alert thresholds	<p>Yellow warning: > 80 %</p> <p>Red critical: > 90 %</p>
Recommended response	<p>If this measurement meets or exceeds 80% with linear growth, increase the value of <code>max_connections</code>.</p> <p>If this measurement meets or exceeds 80% with exponential growth, monitor app usage to ensure everything is behaving as expected.</p> <p>When approaching 100% of max connections, apps may be experiencing times when they cannot connect to the database. The connections/second for a service instance vary based on app instances and app utilization.</p>

Questions

/p.mysql/performance/questions	
Description	<p>The rate of statements executed by the server, shown as queries per second.</p> <p>Use: The server should always be processing some queries, if just as part of the internal automation.</p> <p>Origin: Doppler/Firehose Type: count Frequency: 30 s</p>
Recommended measurement	Average over last 2 minutes
Recommended alert thresholds	<p>Yellow warning: 0 for 90 s</p> <p>Red critical: 0 for 120 s</p>
	Investigate the MySQL server logs, such as the audit log, to understand why query rate changed and determine

Recommended response	appropriate action.
----------------------	---------------------

BOSH System Health Metrics

The BOSH layer that underlies PCF generates `healthmonitor` metrics for all VMs in the deployment. However, these metrics are not included in the Loggregator Firehose by default. To send BOSH HM metrics through the Firehose, install the open-source [HM Forwarder](#).

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

Persistent Disk

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

RAM

system.mem.percent	
Description	<p>RAM being consumed by the MySQL service instance.</p> <p>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.</p> <p>Origin: JMX Bridge or BOSH HM</p> <p>Type: percentage</p> <p>Frequency: 60 s (default)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	<p>Yellow warning: > 95</p> <p>Red critical: > 99</p>
Recommended response	Update the service instance to a plan with more RAM.

CPU

system.cpu.percent	
Description	<p>CPU time being consumed by the MySQL service.</p> <p>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.</p> <p>Origin: JMX Bridge or BOSH HM</p> <p>Type: percent</p>

	Frequency: 60 s (default)
Recommended measurement	Average over 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 plan with larger CPU capacity.

All MySQL Metrics

In addition to the above KPIs, the MySQL service emits the followings metrics that can be used for monitoring and alerting.

Data Source	Description	Metric Unit
/p.mysql/available	Indicates if the local database server is available and responding.	boolean
/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 number of data reads.	reads/second
/p.mysql/innodb/data_writes	The number of data writes.	writes/second
/p.mysql/innodb/mutex_os_waits	The number of mutex OS waits.	events/second
/p.mysql/innodb/mutex_spin_rounds	The number of mutex spin rounds.	events/second
/p.mysql/innodb/mutex_spin_waits	The number of mutex spin waits.	events/second
/p.mysql/innodb/os_log_fsyncs	The number 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 number 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 number of delete statements.	queries/second
/p.mysql/performance/com_delete_multi	The number of delete-multi statements.	queries/second
/p.mysql/performance/com_insert	The number of insert statements.	query/second
/p.mysql/performance/com_insert_select	The number of insert-select statements.	queries/second
/p.mysql/performance/com_replace_select	The number of replace-select statements.	queries/second
/p.mysql/performance/com_select	The number of select statements.	queries/second
/p.mysql/performance/com_update	The number of update statements.	queries/second
/p.mysql/performance/com_update_multi	The number of update-multi statements.	queries/second
/p.mysql/performance/created_tmp_disk_tables	The number of internal on-disk temporary tables created each second by the server while executing statements.	table/second
/p.mysql/performance/created_tmp_files	The number of temporary files created each second.	files/second
	The number of internal temporary tables created each second by the server while	

Data Source	Description	Metric Unit
/p.mysql/performance/created_tmp_tables	Executing statements.	tables/second
/p.mysql/performance/cpu_utilization_percent	The percent of the CPU in use by all processes on the MySQL node.	percent
/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 number of query cache hits.	hits/second
/p.mysql/performance/queries	The number of statements executed by the server, excluding <code>COM_PING</code> and <code>COM_STATISTICS</code> .	count
/p.mysql/performance/queries_delta	The change in the <code>/performance/queries</code> metric since the last time it was emitted.	integer greater than or equal to zero
/p.mysql/performance/questions	The number of statements executed by the server.	queries/second
/p.mysql/performance/slow_queries	The number of slow queries.	queries/second
/p.mysql/performance/table_locks_waited	The total number of times that a request for a table lock could not be granted immediately and a wait was needed.	number
/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/user_time	Percentage of CPU time spent in user space by MySQL.	percent
/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 <code>.frm</code> files that have been cached.	integer
/p.mysql/rpl_semi_sync_master_no_tx	The number of transactions committed without follower acknowledgement.	integer
/p.mysql/rpl_semi_sync_master_tx_avg_wait_time	The average time the leader has waited for the follower to accept transactions.	microseconds
/p.mysql/rpl_semi_sync_master_wait_sessions	The current number of connections waiting for a semi-sync commit.	integer
/p.mysql/variables/read_only	Whether the server is in read-only mode	boolean

Using Leader-Follower Topology for Availability

This topic describes how to use the leader-follower topology for the MySQL service and how to trigger a failover of apps from the leader to the follower.

You can maintain availability of your MySQL databases by keeping two copies of your database, one on a leader (master) instance and another on a follower (slave) instance. If the leader instance fails due to hardware or software issues or during a period of planned maintenance, you can initiate a failover of apps to the follower instance. This helps minimize downtime for apps that bind to your MySQL databases.

For more information, see:

- [About Leader-Follower](#)
- [Events that Interrupt Service](#)

Leader-Follower Setup and Failover Process

The following are the high-level steps for setting up and using a leader-follower topology, and for initiating a failover.

Leader-Follower Setup

1. The operator configures a leader-follower plan. See [Configure a Leader-Follower Service Plan](#) below.
2. Developers create an instance from the leader-follower plan. Doing this automatically deploys two MySQL VMs in a leader-follower topology. See [Create a Service Instance](#).
3. Developers bind apps to the leader in the leader-follower plan instance, using the leader's IP address. Data is written to the leader and asynchronously replicated on the follower. See [Bind a Service Instance to Your App](#).

Failover Process

1. **Promotion:** If the leader instance fails or the operator wants to take the leader offline for maintenance, the operator promotes the follower instance to the new leader.
2. **Failover:** After a promotion, the developer unbinds the app from the leader-follower service instance and then rebinds it in order to fail the app over to the new leader.

See [Trigger a Failover](#) below.

Configure a Leader-Follower Service Plan

You can configure up to five leader-follower service plans.

After you configure the leader-follower plan, it is available in the Marketplace. When a developer creates a leader-follower service instance, a leader VM is automatically deployed in one availability zone (AZ), and a follower VM is deployed in another AZ.

To configure a leader-follower service plan, do the procedure in [Configure Service Plans](#) and select **Multi-node deployment**.

Monitor Leader-Follower Instances

In order to decide if a failover is needed, you must monitor leader-follower instances.

In monitoring the leader-follower VMs, pay attention to the following metrics:

- `/p.mysql/available`: This metric records whether the MySQL VMs are responding to requests. Values are either 1 (available) or 0 (not available).
- `/p.mysql/follower_seconds_since_leader_heartbeat`: Whenever the leader VM emits a heartbeat, the heartbeat is written to the leader database and replicated to the follower. This metric measures the number of seconds that elapses between the leader heartbeat and the replication of the heartbeat in the follower database, so you can determine how far behind the follower is from the leader. Normal values for this metric depend on your apps.

- `/p.mysql/follower_seconds_behind_master`: The follower VM copies database writes from the leader VM, but takes time to apply them to its own database. This metric measures how far behind the follower VM is in applying these writes. For example, a follower VM may have copied writes from the leader VM that are timestamped up to 4:00pm, but it has only applied writes up to 1:00pm. Normal values for this metric depend on your apps.

For more information, see [KPIs for MySQL Service Instances](#) in *Monitoring and KPIs*.

Errands Used in Leader-Follower Failover

This section describes the errands used in the [Trigger a Failover](#) procedures below. You can also use these errands to create custom failover scripts.

Operators use the BOSH CLI to run the errands on the deployment for a particular leader-follower service instance.

The errands used in the failover procedures are:

- **configure-leader-follower:**
 - Configures replication on the follower and ensures the leader is writable
 - Runs after every create or update of a leader-follower instance
 - Fails and alerts operators, via BOSH errand output, if the service instance is in a bad state
- **make-read-only:**
 - Demotes a leader VM to a follower
 - Sets the VM to read-only, and guarantees that apps no longer write to this VM
 - Relays all in-flight transactions on this former leader VM to the follower VM, if it is accessible
- **make-leader:**
 - Promotes a follower VM to a leader
 - Removes replication configuration from the VM, waits for all transactions to be applied to the VM, and sets the VM as writable
 - Fails if the original leader is still writeable to protect against data divergence

Trigger a Failover

You might want to trigger a failover in the following scenarios:

- You want to take the leader VM down to do planned maintenance.
- The performance of the leader VM degrades.
- The leader VM fails unexpectedly.
- The AZ where the leader VM is located goes offline unexpectedly.

To trigger a failover, follow these procedures:

- [Retrieve Information](#)
- [Promote the Follower](#)
- [Clean Up Former Leader VM \(Optional\)](#)
- [Configure the New Follower](#)
- [Unbind and Rebind the App](#)

Retrieve Information

Perform the following steps to retrieve the information necessary for stopping the leader and promoting the follower:

1. Target the Cloud Controller of your PCF deployment with the Cloud Foundry Command Line Interface (cf CLI). For example:

```
$ cf api api.pcf.example.com
```

2. Log in:

```
cf login
```

3. Enter the org and space where the leader-follower service instance is located.
4. Retrieve the GUID of the service instance by running the following command:

```
cf service SERVICE-INSTANCE --guid
```

Where `SERVICE-INSTANCE` is the name of the leader-follower service instance.

For example:

```
$ cf service my-lf-instance --guid
82ddc607-710a-404e-b1b8-a7e3ea7ec063
```

If you do not know the name of the service instance, you can list service instances in the space with `cf services`.

5. Perform the steps in [Gather Credential and IP Address Information](#) and [SSH into Ops Manager](#) of *Advanced Troubleshooting with the BOSH CLI* to SSH into the Ops Manager VM.
6. From the Ops Manager VM, log in to your BOSH Director with the BOSH CLI. See [Log in to the BOSH Director](#) in *Advanced Troubleshooting with the BOSH CLI*.
7. Use the BOSH CLI to run the `inspect` errand with the following command:

```
bosh2 -d service-instance_GUID run-errand inspect
```

Where `GUID` is the GUID of the leader-follower service instance retrieved above.

For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
run-errand inspect
```

8. Examine the output and locate the information about the leader-follower MySQL VMs:

```
Instance mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0
Exit Code 0
Stdout 2018/04/03 18:08:46 Started executing command: inspect
2018/04/03 18:08:46
IP Address: 10.0.8.11
Role: leader
Read Only: false
Replication Configured: false
Replication Mode: async
Has Data: true
GTID Executed: 82ddc607-710a-404e-b1b8-a7e3ea7ec063:1-18
2018/04/03 18:08:46 Successfully executed command: inspect
Stderr -

Instance mysql/37e4b6bc-2ed6-4bd2-84d1-e59a91f5e7f8
Exit Code 0
Stdout 2018/04/03 18:08:46 Started executing command: inspect
2018/04/03 18:08:46
IP Address: 10.0.8.10
Role: follower
Read Only: true
Replication Configured: true
Replication Mode: async
Has Data: true
GTID Executed: 82ddc607-710a-404e-b1b8-a7e3ea7ec063:1-18
2018/04/03 18:08:46 Successfully executed command: inspect
Stderr -
```

9. Identify the instance marked `Role: leader`. This is the leader VM. Record the index, which is the value for `Instance` after `mysql/`. In the above output, the index of the leader VM is `ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0`.
10. Record the index of the other instance, which is the follower VM. In the above output, the index of the follower VM is `37e4b6bc-2ed6-4bd2-84d1-e59a91f5e7f8`.
11. If you still have access to the AZ where the leader VM is located, use the BOSH CLI to determine if the leader VM is in the AZ you want to take offline.

For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
instances
Deployment 'service-instance_f378ec82-61a4-4e66-8ed9-889c7cf5342f'

Instance                Process State AZ      IPs
mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0 failing    us-central1-f 10.0.8.11
mysql/37e4b6bc-2ed6-4bd2-84d1-e59a91f5e7f8 running    us-central1-a 10.0.8.10
2 instances
```



Note: The leader VM might not display its status as `failing` if you are performing planned maintenance.

Examine the output to determine if the leader VM is in the AZ you want to take offline.

Promote the Follower

Perform the following steps to stop the leader VM and promote the follower VM to the new leader:

1. Stop any data from being written to the leader VM by setting it to read only. Run the following command:

```
bosh2 -d service-instance_GUID \
run-errand make-read-only \
--instance=mysql/INDEX
```

Where:

- `GUID` : This is the GUID of the leader-follower service instance retrieved above.
- `INDEX` : This is the index of the leader VM retrieved above.

For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
run-errand make-read-only \
--instance=mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0
```

2. If you still have access to the AZ where the leader VM is located, stop the leader VM, using the same values as above. For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
stop mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0
```

3. Set the follower VM as writable:

- If you are using PCF v2.0 and above, run the `make-leader` errand, using the index of the follower VM retrieved above. For example:

```
$ bosh2 -d service-instance_82dc607-710a-404e-b1b8-a7e3ea7ec063 \
run-errand make-leader \
--instance=mysql/37e4b6bc-2ed6-4bd2-84d1-e59a91f5e7f8
```

If this command returns an error, re-run it until the follower VM has completed applying the transactions.

- If you are using PCF v1.12, use `bosh2 ssh` to run the errand, using the index of the follower VM retrieved above. For example:

```
$ bosh2 -d service-instance_82dc607-710a-404e-b1b8-a7e3ea7ec063 \
ssh \
mysql/37e4b6bc-2ed6-4bd2-84d1-e59a91f5e7f8 \
-c "sudo /var/vcap/jobs/make-leader/bin/run"
```

At this point, a single instance is working but leader-follower replication has not yet been restored. If you want to fail your app over to a single instance instead of restoring leader-follower, skip to [Unbind and Rebind the App](#).

If you are triggering a failover in response to the AZ of the leader VM going offline, you can fail your app over to a single instance by performing the steps in [Unbind and Rebind the App](#). But in order to restore leader-follower, you must regain access to the AZ where your leader VM is located before performing the steps in [Clean Up Former Leader VM \(Optional\)](#) and [Configure the New Follower](#).

Clean Up Former Leader VM (Optional)

If you are triggering a failover in response to a failing leader VM, perform these steps to clean up the former leader VM:

1. Disable resurrection, specifying the same deployment as above. For example:

```
$ bosh2 update-resurrection off
```

2. Use the BOSH CLI to retrieve the CID of the failing former leader VM. For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 instances \
--details \
--failing \
--column="VM CID" \
--json
```

3. Use the BOSH CLI to retrieve the disk CID of the failing former leader VM. For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 instances \
--details \
--failing \
--column="Disk CIDs" \
--json
```

4. Delete the failing former leader VM. Run the following command:

```
bosh2 -d service-instance_GUID delete-vm vm-CID
```

Where:

- **GUID** : This is the GUID of the leader-follower service instance retrieved above.
- **CID** : This is the CID of the failing former leader VM retrieved above.

For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
delete-vm i-1db9ede6
```

5. Orphan the disk of the failing former leader VM. Run the following command:

```
bosh2 -d service-instance_GUID orphan-disk DISK-CID
```

Where:

- **GUID** : This is the GUID of the leader-follower service instance retrieved above.
- **DISK-CID** : This is the disk CID of the failing former leader VM retrieved above.

For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
orphan-disk b-1db9ede6
```

Orphaning a disk rather than deleting it preserves it for possible recovery. After performing recovery operations, you can reattach the disk to a VM. BOSH deletes orphaned disks after five days by default.

Configure the New Follower

Perform the following steps to restart the former leader VM and configure it as the new follower:

1. Recreate the former leader VM. Run the following command:

```
bosh2 -d service-instance_GUID \
recreate \
mysql/INDEX
```

Where:

- `GUID`: This is the GUID of the leader-follower service instance retrieved above.
- `INDEX`: This is the index of the former leader VM that you are recreating.

For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
recreate \
mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd01.
```

2. Use the `configure-leader-follower` errand to set the former leader VM as a follower, using the same values as above. For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
run-errand configure-leader-follower \
--instance=mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0
```

3. Use the BOSH CLI to run the `inspect` errand on the deployment, using the same value as above. For example:

```
$ bosh2 -d service-instance_82ddc607-710a-404e-b1b8-a7e3ea7ec063 \
run-errand inspect
```

If the output displays one instance marked `Role: leader` and another instance marked `Role: follower`, then leader-follower replication and high availability are resumed. The deployment should be in its original, working state. You may turn resurrection back on if desired.

Unbind and Rebind the App

To fail their apps over to the new leader VM, your developers must perform the following steps to bind and rebind their apps to the leader-follower service instance:

1. Unbind the app from the leader-follower service instance. Run the following command:

```
cf unbind-service APP SERVICE-INSTANCE
```

Where:

- `APP`: This is the name of the app bound to the leader-follower service instance.
- `SERVICE-INSTANCE`: This is the name of the leader-follower service instance.

For example:

```
$ cf unbind-service my-app my-lf-instance
```

2. Rebind the app to the leader-follower service instance, using the same values as above. For example:

```
$ cf bind-service my-app my-lf-instance
```

3. Restage the app. For example:

```
$ cf restage my-app
```

Events that Interrupt Service

This topic explains events in the lifecycle of a MySQL for Pivotal Cloud Foundry (PCF) service instance that may cause temporary service interruptions.

Stemcell or Service Update

An operator updates a stemcell version or their version of MySQL for PCF.

- **Impact:** Apps lose access to the MySQL service while PCF updates the service instance they are bound to. The service should resume within 10-15 minutes.
- **Required Actions:** None. If the update deploys successfully, apps reconnect automatically.

Plan Change

A developer changes their service instance to provide a different service plan, using `cf update-service` or Apps Manager.

- **Impact:** Apps lose access to the MySQL service while PCF updates the service instance they are bound to. The service should resume within 10-15 minutes.
- **Required Actions:** None. If the plan change deploys successfully, apps reconnect automatically.

VM Process Failure

A process, like the MySQL server, crashes on the service instance VM.

- **Impact:**
 - BOSH (monit) brings the process back automatically.
 - Depending on the process and what it was doing, the service may experience 60-120 seconds of downtime.
 - Until the process resumes, apps may be unable to use MySQL, metrics or logging may stop, and other features may be interrupted.
- **Required Actions:** None. If the process resumes cleanly and without manual intervention, apps reconnect automatically.

VM Failure

A MySQL for PCF VM fails and goes offline due to either a virtualization problem or a host hardware problem.

- **Impact:**
 - If the [BOSH Resurrector](#) is enabled (recommended), BOSH should detect the failure, recreate the VM, and reattach the same persistent disk and IP address.
 - Downtime largely depends on how quickly the Resurrector notices, usually 1-2 minutes, and how long it takes the IaaS to create a replacement VM.
 - If the Resurrector is not enabled, some IaaS such as vSphere have similar resurrection or HA features.
 - Apps cannot connect to MySQL until the VM is recreated and the MySQL server process is resumed.
 - Based on prior experience with BOSH Resurrector, typical downtime is 8-10 minutes.
- **Required Actions:**
 - If the VM is part of a leader-follower pair, when the VM comes back, it is read-only. Therefore, run the `configure-leader-follower` errand to ensure the leader VM is writable. For more information, see [Leader-Follower Errands](#).
 - If the VM is not part of a leader-follower pair, when the VM comes back, no further action is required for the app developer to continue operations.

AZ Failure

An Availability Zone (AZ) goes offline entirely or loses connectivity to other AZs (net split). This causes service interruption in multi-AZ PCF deployments where [Diego](#) has placed multiple instances of a MySQL-using app in different AZs.

- **Impact:**
 - Some app instances may still be able to connect and continue operating.
 - App instances in the other AZs will not be able to connect.
 - Downtime: Unknown
- **Required Actions:** Recovery of the app / database connection should be automatic. Depending on the app, manual intervention may be required to check data consistency.


Region Failure

- **Example:** An entire region fails, bringing PCF platform components offline.
- **Impact:**
 - The entire PCF platform needs to be brought back up manually.
 - Downtime: Unknown
- **Required Actions:** Each service instance may need to be [restored individually](#) depending on the restored state of the platform.

Upgrading MySQL for PCF

This topic explains how to upgrade the MySQL for Pivotal Cloud Foundry (PCF) service and existing service instances. It also explains the service interruptions that can result from service changes and upgrades and from failures at the process, VM, and IaaS level.

Upgrade MySQL for PCF

 **Note:** Before upgrading to the next version of MySQL for PCF, run the `upgrade-all-service-instances` errand to make sure that all your service instances are upgraded to your current version.
For more information, see [Upgrade All Service Instances](#).

To upgrade the MySQL for PCF service, you follow the same Ops Manager process that you use to install the service for the first time. Your configuration settings migrate to the new version automatically. To perform an upgrade:

1. Review the [Release Notes](#) for the version you are upgrading to.
2. Download the desired version of the product from [Pivotal Network](#).
3. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.
4. Under the **Import a Product** button, click + next to **MySQL for PCF**. This adds the tile to your staging area.
5. Click the newly-added **MySQL for PCF** tile to review its configuration panes. Click **Save** on any panes where you make changes.
6. Return to the Installation Dashboard and click **Apply Changes**.

Upgrading the MySQL for PCF service and service instances can temporarily interrupt the service, as described [below](#).

Upgrade MySQL Instances

After upgrading the MySQL for PCF service, operators must upgrade existing service instances to be in sync with the new version of the service. Developers cannot create new bindings to a service instance that has not been upgraded.

To upgrade existing service instances, operators run the following command:

```
bosh -d BROKER-NAME run-errand upgrade-all-service-instances
```

Where `BROKER-NAME` is the deployment name of the broker.

Service Interruptions

Service changes and upgrades and failures at the process, VM, and IaaS level can cause outages in the MySQL for PCF service, as described below.

Read this section if:

- You are experiencing a service interruption and are wondering why.
- You are planning to update or change a service instance and want to know if it might cause a service interruption.

Stemcell or Service Update

An operator updates a stemcell version or their version of MySQL for PCF.

- **Impact:** Apps lose access to the MySQL service while PCF updates the service instance they are bound to. The service should resume within 10–15 minutes.
- **Required Actions:** None. If the update deploys successfully, apps reconnect automatically.

Plan Change

A developer changes their service instance to provide a different service plan, using `cf update-service` or Apps Manager.

- **Impact:** Apps lose access to the MySQL service while PCF updates the service instance they are bound to. The service should resume within 10–15 minutes.
- **Required Actions:** None. If the plan change deploys successfully, apps reconnect automatically.

Troubleshooting MySQL for PCF

This topic provides operators with basic instructions for troubleshooting on-demand MySQL for PCF.

For information on temporary MySQL for PCF service interruptions, see [Service Interruptions](#).

Troubleshooting Errors

This section provides information on how to troubleshooting specific errors or error messages.

Failed Install

1. Certificate issues: The on-demand broker (ODB) requires valid certificates. Ensure that your certificates are valid and generate new ones if necessary. To generate new certificates, contact [Pivotal Support](#).
2. Deploy fails: Deploys can fail for a variety of reasons. View the logs using Ops Manager to determine why the deploy is failing.
3. [Networking problems](#):
 - Cloud Foundry cannot reach the MySQL for PCF service broker
 - Cloud Foundry cannot reach the service instances
 - The service network cannot access the BOSH director
4. [Register broker errand](#) fails.
5. The smoke test errand fails.
6. Resource sizing issues: These occur when the resource sizes selected for a given plan are less than the MySQL for PCF service requires to function. Check your resource configuration in Ops Manager and ensure that the configuration matches that recommended by the service.
7. Other service-specific issues.

Cannot Create or Delete Service Instances

If developers report errors such as:

```
Instance provisioning failed: There was a problem completing your request. Please contact your operations team providing the following information: service: redis-acceptance, service-instance-g
```

Follow these steps:

1. If the BOSH error shows a problem with the deployment manifest, open the manifest in a text editor to inspect it.
2. To continue troubleshooting, [Log in to BOSH](#) and target the MySQL for PCF service instance using the instructions on [parsing a Cloud Foundry error message](#).
3. Retrieve the BOSH task ID from the error message and run the following command:

```
bosh task TASK-ID
```

4. If you need more information, [access the broker logs](#) and use the `broker-request-id` from the error message above to search the logs for more information. Check for:
 - [Authentication errors](#)
 - [Network errors](#)

- [Quota errors](#)

Broker Request Timeouts

If developers report errors such as:

```
Server error, status code: 504, error code: 10001, message: The request to the service broker timed out: https://BROKER-URL/v2/service_instances/e34046d3-2379-40d0-a318-d54fc7a5b13f/ser
```

Follow these steps:

1. Confirm that Cloud Foundry (CF) is [connected to the service broker](#).
2. Check the BOSH queue size:
 - a. Log into BOSH as an admin.
 - b. Run `bosh tasks`.
3. If there are a large number of queued tasks, the system may be under too much load. BOSH is configured with two workers and one status worker, which may not be sufficient resources for the level of load. Advise app developers to try again once the system is under less load.

Cannot Bind to or Unbind from Service Instances

Instance Does Not Exist

If developers report errors such as:

```
Server error, status code: 502, error code: 10001, message: Service broker error: instance does not exist
```

Follow these steps:

1. Confirm that the MySQL for PCF service instance exists in BOSH and obtain the GUID CF by running:

```
cf service MY-INSTANCE --guid
```

2. Using the GUID obtained above, the following BOSH CLI command:

```
bosh -d service-instance_GUID vms
```

If the BOSH deployment is not found, it has been deleted from BOSH. Contact Pivotal support for further assistance.

Other Errors

If developers report errors such as:

```
Server error, status code: 502, error code: 10001, message: Service broker error: There was a problem completing your request. Please contact your operations team providing the following inform
```

To find out the exact issue with the binding process:

1. [Access the service broker logs](#).

2. Search the logs for the `broker-request-id` string listed in the error message above.
3. Contact Pivotal support for further assistance if you are unable to resolve the problem.
4. Check for:
 - [Authentication errors](#)
 - [Network errors](#)

Cannot Connect to a Service Instance

If developers report that their app cannot use service instances that they have successfully created and bound:

Ask the user to send application logs that show the connection error. If the error is originating from the service, then follow MySQL for PCF-specific instructions. If the issue appears to be network-related, then:

1. Check that [application security groups](#) are configured correctly. Access should be configured for the service network that the tile is deployed to.
2. Ensure that the network the Pivotal Application Service (PAS) tile is deployed to has network access to the service network. You can find the network definition for this service network in the BOSH Director tile.
3. In Ops Manager go into the service tile and see the service network that is configured in the networks tab.
4. In Ops Manager go into the PAS tile and see the network it is assigned to. Make sure that these networks can access each other.

Service instances can also become temporarily inaccessible during upgrades and VM or network failures. See [Service Interruptions](#) for more information.

Upgrade All Instances Fails

If the `upgrade-all-service-instances` errand fails, look at the errand output in the Ops Manager log.

If an instance fails to upgrade, debug and fix it before running the errand again to prevent any failure issues from spreading to other on-demand instances.

Once the Ops Manager log no longer lists the deployment as `failing`, [re-run the errand](#) to upgrade the rest of the instances.

Missing Logs and Metrics

If no logs are being emitted by the on-demand broker, check that your syslog forwarding address is correct in Ops Manager.

1. Ensure you have configured syslog for the tile.
2. Ensure that you have network connectivity between the networks that the tile is using and the syslog destination. If the destination is external, you need to use the [public ip](#) VM extension feature available in your Ops Manager tile configuration settings.
3. Verify that the Firehose is emitting metrics:
 - a. Install the `cf nozzle` plugin. For instructions, see the [firehose plugin](#) GitHub repository.
 - b. To find logs from your service in the `cf nozzle` output, run the following:

```
cf nozzle -f ValueMetric | grep --line-buffered "on-demand-broker/MY-SERVICE"
```

If no metrics appear within five minutes, verify that the broker network has access to the Loggregator system on all required ports.

[Contact Pivotal support](#) if you are unable to resolve the issue.

Unable to Determine Leader and Follower (Errand Error)

This problem happens when the `configure-leader-follower` errand fails because it cannot determine the VM roles.

Symptom

The `configure-leader-follower` errand exits with `1` and the errand logs contain the following:

```
$ Unable to determine leader and follower based on transaction history.
```

Explanation

Something has happened to the instances, such as a failure or manual intervention. As a result, there is not enough information available to determine the correct state and topology without operator intervention to resolve the issue.

Solution

Use the `inspect` errand to determine which instance should be the leader. Then, using the [orchestration](#) errands and backup/restore, you can put the service instance into a safe topology, and then rerun the `configure-leader-follower` errand. This is shown in the example below. This example shows one outcome that the `inspect` errand can return:

1. Use the `inspect` errand to retrieve relevant information about the two VMs:

```
$ bosh2 -e my-env -d my-dep run-errand inspect
[...]
Instance mysql/4ecad54b-0704-47eb-8eef-eb228cab9724
Exit Code 0
Stdout -
Stderr 2017/12/11 18:25:54 Started executing command: inspect
2017/12/11 18:25:54 Started GET https://127.0.0.1:8443/status
2017/12/11 18:25:54
Has Data: false
Read Only: true
GTID Executed: 1d774323-de9e-11e7-be01-42010a001014:1-25
Replication Configured: false

Instance mysql/e0b94ade-0114-4d49-a929-ce1616d8beda
Exit Code 0
Stdout -
Stderr 2017/12/11 18:25:54 Started executing command: inspect
2017/12/11 18:25:54 Started GET https://127.0.0.1:8443/status
2017/12/11 18:25:54
Has Data: true
Read Only: true
GTID Executed: 1d774323-de9e-11e7-be01-42010a001014:1-25
Replication Configured: true

2 errand(s)

Succeeded
```

In the above scenario, the first instance is missing data but does not have replication configured. The second instance has data, and also has replication configured. The instructions below resolve this by copying data to the first instance, and resuming replication.

2. Take a backup of the second instance using the [Manual Backup](#) steps.
3. Restore the backup artifact to the first instance using the [Manual Restore](#) steps.

At this point, the instances have equivalent data.

4. Run the `configure-leader-follower` errand to reconfigure replication:

```
bosh2 -e ENVIRONMENT -d DEPLOYMENT run-errand configure-leader-follower --instance=mysql/GUID-OF-LEADER
```

```
$ bosh2 -e my-env -d my-dep \
run-errand configure-leader-follower \
--instance=mysql/4ecad54b-0704-47eb-8eef-eb228cab9724
```

Both Leader and Follower Instances Are Writable (Errand Error)

This problem happens when the `configure-leader-follower` errand fails because both VMs are writable and the VMs might hold differing data.

Symptom

The `configure-leader-follower` errand exits with `1` and the errand logs contain the following:

```
$ Both mysql instances are writable. Please ensure no divergent data and set one instance to read-only mode.
```

Explanation

MySQL for PCF tries to ensure that there is only one writable instance of the leader-follower pair at any given time. However, in certain situations, such as network partitions, or manual intervention outside of the provided bosh errands, it is possible for both instances to be writable. The service instances remain in this state until an operator resolves the issue to ensure that the correct instance is promoted and reduce the potential for data divergence.

Solution

1. Use the `inspect` errand to retrieve the GTID Executed set for each VM:

```
$ bosh2 -e my-env -d my-dep run-errand inspect
[...]
Instance mysql/4ecad54b-0704-47eb-8eef-eb228cab9724
Exit Code 0
Stdout -
Stderr 2017/12/11 18:25:54 Started executing command: inspect
2017/12/11 18:25:54 Started GET https://127.0.0.1:8443/status
2017/12/11 18:25:54
Has Data: true
Read Only: false
GTID Executed: 1d774323-de9e-11e7-be01-42010a001014:1-23
Replication Configured: false

Instance mysql/e0b94ade-0114-4d49-a929-ce1616d8beda
Exit Code 0
Stdout -
Stderr 2017/12/11 18:25:54 Started executing command: inspect
2017/12/11 18:25:54 Started GET https://127.0.0.1:8443/status
2017/12/11 18:25:54
Has Data: true
Read Only: false
GTID Executed: 1d774323-de9e-11e7-be01-42010a001014:1-25
Replication Configured: false

2 errand(s)

Succeeded
```

If the GTID Executed sets for both instances are the same, continue to Step 2. If they are different, continue to Step 4.

2. Look at the value of GTID Executed for both instances.
 - If the range after the GUID is equivalent, either instance can be made read-only, as described in Step 3.
 - If one instance has a range that is a subset of the other, the instance with the subset should be made read-only, as described in Step 3.
3. Based on the information you gathered in the step above, run the `make-read-only` errand to make the appropriate instance read-only:

```
bosh2 -e ENVIRONMENT -d DEPLOYMENT run-errand make-read-only --instance=mysql/MYSQL-SUBSET-INSTANCE
```

```
$ bosh2 -e my-env -d my-dep \
run-errand make-read-only \
--instance=mysql/e0b94ade-0114-4d49-a929-ce1616d8beda
[...]
Succeeded
```

4. If the GTID Executed sets are neither equivalent nor subsets, data has diverged and you must determine what data has diverged as part of the

procedure below:

- a. Use the `make-read-only` errand to set both instances to read-only to prevent further data divergence.

```
bosh2 -e ENVIRONMENT -d DEPLOYMENT run-errand make-read-only --instance=mysql/MYSQL-INSTANCE
```

```
$ bosh2 -e my-env -d my-dep \
  run-errand make-read-only \
  --instance=mysql/c0b94ade-0114-4d49-a929-ce1616d8beda
[...]
Succeeded
```

- b. Take a backup of both instances using the [Manual Backup](#) steps.
- c. Manually inspect the data on each instance to determine the discrepancies and put the data on the instance that is further ahead—this instance has the higher GTID Executed set, and will be the new leader.
- d. Migrate all appropriate data to the new leader instance.
- e. After putting all data on the leader, ssh onto the follower: `bosh2 -e ENVIRONMENT -d DEPLOYMENT ssh mysql/GUID-OF-FOLLOWER`

```
$ bosh2 -e my-env -d my-dep ssh mysql/c0b94ade-0114-4d49-a929-ce1616d8beda
```

- f. Become root with the command `sudo su`.
- g. Stop the mysql process with the command `monit stop mysql`.
- h. Delete the data directory of the follower with the command `rm -rf /var/vcap/store/mysql`.
- i. Start the mysql process with the command `monit start mysql`.
- j. Use the `configure-leader-follower` errand to copy the leader's data to the follower and resume replication:

```
bosh2 -e ENVIRONMENT -d DEPLOYMENT run-errand configure-leader-follower --instance=mysql/GUID-OF-LEADER
```

```
$ bosh2 -e my-env -d my-dep \
  run-errand configure-leader-follower \
  --instance=mysql/4ecad54b-0704-47eb-8eef-eb228cab9724
```

Both Leader and Follower Instances Are Read-Only

In a leader-follower topology, the leader VM is writable and the follower VM is read-only. However if both VMs are read only, apps cannot write to the database.

Symptom

Developers report that apps cannot write to the database.

Explanation

This problem happens if the leader VM fails and the BOSH Resurrector is enabled. When the leader is resurrected, it is set as read-only.

Solution

1. Use the `inspect` errand to confirm that both VMs are in a read-only state: `bosh2 -e ENVIRONMENT -d DEPLOYMENT run-errand inspect`
2. Examine the output and locate the information about the leader-follower MySQL VMs:

```
Instance mysql/4eexample54b-0704-47eb-8eef-eb2example724
Exit Code 0
Stdout -
Stderr 2017/12/11 18:25:54 Started executing command: inspect
2017/12/11 18:25:54 Started GET https:999.0.0.1:8443/status
2017/12/11 18:25:54
Has Data: true
Read Only: true
GTID Executed: 1d779999-de9e-11e7-be01-42010a009999:1-23
Replication Configured: true
```

```
Instance mysql/e0exampleade-0114-4d49-a929-cexample8beda
Exit Code 0
Stdout -
Stderr 2017/12/11 18:25:54 Started executing command: inspect
2017/12/11 18:25:54 Started GET https:999.0.0.1:8443/status
2017/12/11 18:25:54
Has Data: true
Read Only: true
GTID Executed: 1d779999-de9e-11e7-be01-42010a009999:1-25
Replication Configured: false
```

2 errand(s)

Succeeded

- If Read Only is set to `true` for both VMs, make the leader writable using the following command:

```
bosh2 -e ENVIRONMENT -d DEPLOYMENT run-errand configure-leader-follower --instance=mysql/GUID-OF-LEADER
```

For example, if the second instance above is the leader:

```
$ bosh2 -e my-env -d my-dep \
run-errand configure-leader-follower \
--instance=mysql/e0exampleade-0114-4d49-a929-cexample8beda
```

Persistent Disk is Full

If your persistent disk is full, apps become inoperable. In this state, read, write, and Cloud Foundry Command-Line Interface (cf CLI) operations do not work.

Symptom

Developers report that read, write, and cf CLI operations do not work. Developers cannot upgrade to a larger MySQL service plan to free up disk space.

Explanation

This problem happens if your persistent disk is full. When you use the BOSH CLI to target your deployment, you see that instances are at 100% persistent disk usage.

Available disk space can be increased by deleting audit log files. After deleting audit logs, you can then upgrade to a larger MySQL service plan.

Solution

- To retrieve and record the GUID of your service instance , run the following command:

```
cf service SERVICE-INSTANCE-NAME --guid
```

Where `SERVICE-INSTANCE-NAME` is the name of your service instance.

For example:

```
$ cf service my-service-instance --guid
12345678-90ab-cdef-1234-567890abcdef
```

If you do not know the name of your service instance, you can list service instances in the space with `cf services`.

- To confirm that your persistent disk usage is at 100%, run the following command:

```
bosh -d service-instance_SERVICE-INSTANCE-GUID instances --vitals
```

Where `SERVICE-INSTANCE-GUID` is the GUID you recorded in the above step.

For example:

```
$bosh -d service-instance_12345678-90ab-cdef-1234-567890abcdef instances --vitals
Using environment 'https://10.0.0.6:25555' as client 'admin'

Task 19243: Done

Deployment 'service-instance_12345678-90ab-cdef-1234-567890abcdef'

Instance          Process State AZ  IPs          VM Created At      Uptime          Load          CPU          CPU          CPU          CPU          Memory          Swap          System          Ephemeral
mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0 running      z3  10.0.18.20 Wed Sep 12 22:01:44 UTC 2018 35d 20h 54m 17s 0.02,0.03,0.00 - 10.2% 0.4% 0.2% 14% (1.1 GB)

1 instances

Succeeded
```

- To retrieve and record the instance ID of your service instance, follow the procedure below that corresponds with your VM topology.

- If you are using single node MySQL VMs, to retrieve and record the instance ID, run the following command:

```
bosh -d service-instance_SERVICE-INSTANCE-GUID instances
```

Where `SERVICE-INSTANCE-GUID` is the GUID you recorded in the step one.

For example:

```
$ bosh -d service-instance_12345678-90ab-cdef-1234-567890abcdef
instances
Using environment '34.237.123.534' as client 'admin'

Task 204: Done

Deployment 'service-instance_12345678-90ab-cdef-1234-567890abcdef'

Instance          Process State AZ  IPs          VM Created At      Uptime          Load          CPU          CPU          CPU          CPU          Memory          Swap          System          Ephemeral
mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0 running      z2  10.244.17.3

1 instances

Succeeded
```

The instance ID is the value for `Instance` after `mysql/`.

In the above output, the instance ID of the leader VM is `d15419ba-fc6c-4013-b056-19f91c6b0f1d`.

- If you are using leader-follower MySQL VMs, to retrieve and record the leader instance ID, run the following command:

```
bosh -d service-instance_SERVICE-INSTANCE-GUID run-errand inspect
```

Where `SERVICE-INSTANCE-GUID` is the GUID you recorded in the step one.

For example:

```
$ bosh -d service-instance_12345678-90ab-cdef-1234-567890abcdef run-errand inspect
Instance mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0
Exit Code 0
Stdout 2018/04/03 18:08:46 Started executing command: inspect
2018/04/03 18:08:46
IP Address: 10.0.8.11
Role: leader
Read Only: false
Replication Configured: false
Replication Mode: async
Has Data: true
GTID Executed: 82ddc607-710a-404e-b1b8-a7e3ea7ec063:1-18
2018/04/03 18:08:46 Successfully executed command: inspect
Stderr -

Instance mysql/37e4b6bc-2ed6-4bd2-84d1-e59a91f5e7f8
Exit Code 0
Stdout 2018/04/03 18:08:46 Started executing command: inspect
2018/04/03 18:08:46
IP Address: 10.0.8.10
Role: follower
Read Only: true
Replication Configured: true
Replication Mode: async
Has Data: true
GTID Executed: 82ddc607-710a-404e-b1b8-a7e3ea7ec063:1-18
2018/04/03 18:08:46 Successfully executed command: inspect
Stderr -
```

The leader instance ID is the value for `Instance` after `mysql/` corresponding with the instance marked `Role: leader`.

In the above output, the instance ID of the leader VM is `ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0`.

- To BOSH SSH into your service instances, run the following command:

```
bosh -d service-instance_SERVICE-INSTANCE-GUID ssh mysql/INSTANCE-ID
```

Where:

- `SERVICE-INSTANCE-GUID` is the GUID you recorded in the step one.
- `INSTANCE-ID` is the instance ID you recorded in the previous step

For example:

```
$ bosh -d service-instance_12345678-90ab-cdef-1234-567890abcdef ssh mysql/ca0ed8b5-7590-4cde-bba8-7ca2935f2bd0
Using environment 'https://10.0.0.6:25555' as client 'admin'

Using deployment 'service-instance_12345678-90ab-cdef-1234-567890abcdef'

Task 19244. Done
```

- To move to the directory where your audit log files are located, run the following command:

```
cd /var/vcap/sys/log/mysql/mysql-audit-log
```

- Delete at least one audit log file with `rm`

- Update your service instance to a larger plan. For more information, see [Update a Service Instance to a Larger Plan](#).

Troubleshooting Components

This section provides guidance on checking for and fixing issues in on-demand service components.

BOSH Problems

Large BOSH Queue

On-demand service brokers add tasks to the BOSH request queue, which can back up and cause delay under heavy loads. An app developer who requests a new MySQL for PCF instance sees `create in progress` in the Cloud Foundry Command Line Interface (cf CLI) until BOSH processes the queued request.

Ops Manager currently deploys two BOSH workers to process its queue. Future versions of Ops Manager will let users configure the number of BOSH workers.

Configuration

Service instances in failing state


You may have configured a VM / Disk type in tile plan page in Ops Manager that is insufficiently large for the MySQL for PCF service instance to start. See tile-specific guidance on resource requirements.

Authentication

UAA Changes

If you have rotated any UAA user credentials then you may see authentication issues in the service broker logs.

To resolve this, redeploy the MySQL for PCF tile in Ops Manager. This provides the broker with the latest configuration.

 **Note:** You must ensure that any changes to UAA credentials are reflected in the Ops Manager `credentials` tab of the Pivotal Application Service (PAS) tile.

Networking

Common issues with networking include:

Issue	Solution
Latency when connecting to the MySQL for PCF service instance to create or delete a binding.	Try again or improve network performance.
Firewall rules are blocking connections from the MySQL for PCF service broker to the service instance.	Open the MySQL for PCF tile in Ops Manager and check the two networks configured in the Networks pane. Ensure that these networks allow access to each other.
Firewall rules are blocking connections from the service network to the BOSH director network.	Ensure that service instances can access the Director so that the BOSH agents can report in.
Apps cannot access the service network.	Configure Cloud Foundry application security groups to allow runtime access to the service network.
Problems accessing BOSH's UAA or the BOSH director.	Follow network troubleshooting and check that the BOSH director is online

Validate Service Broker Connectivity to Service Instances

To validate connectivity, do the following:

1. To SSH into the MySQL for PCF service broker, run the following command:

```
bosh -d service-instance_GUID ssh
```

2. If no BOSH `task-id` appears in the error message, look in the broker log using the `broker-request-id` from the task.

Validate App Access to Service Instance

Use `cf ssh` to access to the app container, then try connecting to the MySQL for PCF service instance using the binding included in the `VCAP_SERVICES` environment variable.

Quotas

Plan Quota issues

If developers report errors such as:

```
Message: Service broker error: The quota for this service plan has been exceeded.  
Please contact your Operator for help.
```

1. Check your current plan quota.
2. Increase the plan quota.
3. Log into Ops Manager.
4. Reconfigure the quota on the plan page.
5. Deploy the tile.
6. Find who is using the plan quota and take the appropriate action.

Global Quota Issues

If developers report errors such as:

```
Message: Service broker error: The quota for this service has been exceeded.  
Please contact your Operator for help.
```

1. Check your current global quota.
2. Increase the global quota.
3. Log into Ops Manager.
4. Reconfigure the quota on the on-demand settings page.
5. Deploy the tile.

- Find out who is using the quota and take the appropriate action.

Failing Jobs and Unhealthy Instances

To determine whether there is an issue with the MySQL for PCF service deployment, inspect the VMs. To do so, run the following command:

```
bosh -d service-instance_GUID vms --vitals
```

For additional information, run the following command:

```
bosh instances --ps --vitals
```

If the VM is failing, follow the service-specific information. Any unadvised corrective actions (such as running BOSH `restart` on a VM) can cause issues in the service instance.

A failing process or failing VM might come back automatically after a temporary service outage. See [VM Process Failure](#) and [VM Failure](#).

AZ or Region Failure

Failures at the IaaS level, such as Availability Zone (AZ) or region failures, can interrupt service and require manual restoration. See [AZ Failure](#) and [Region Failure](#).

Techniques for Troubleshooting

Instructions on interacting with the on-demand service broker and on-demand service instance BOSH deployments, and on performing general maintenance and housekeeping tasks

Parse a Cloud Foundry (CF) Error Message

Failed operations (create, update, bind, unbind, delete) result in an error message. You can retrieve the error message later by running the cf CLI command `cf service INSTANCE-NAME`.

```
$ cf service myservice

Service instance: myservice
Service: super-db
Bound apps:
Tags:
Plan: dedicated-vm
Description: Dedicated Instance
Documentation url:
Dashboard:

Last Operation
Status: create failed
Message: Instance provisioning failed: There was a problem completing your request.
Please contact your operations team providing the following information:
service: redis-acceptance,
service-instance-guid: ae9e232c-0bd5-4684-af27-1b08b0c70089,
broker-request-id: 63da3a35-24aa-4183-aec6-db8294506bac,
task-id: 442,
operation: create
Started: 2017-03-13T10:16:55Z
Updated: 2017-03-13T10:17:58Z
```

Use the information in the `Message` field to debug further. Provide this information to Pivotal Support when filing a ticket.

The `task-id` field maps to the BOSH task ID. For more information on a failed BOSH task, use the `bosh task TASK-ID`.

The `broker-request-guid` maps to the portion of the On-Demand Broker log containing the failed step. Access the broker log through your syslog

aggregator, or access BOSH logs for the broker by typing `bosh logs broker 0`. If you have more than one broker instance, repeat this process for each instance.

Access Broker and Instance Logs and VMs

Before following the procedures below, log into the [cf CLI](#) and the [BOSH CLI](#).

Access Broker Logs and VM(s)

You can [access logs using Ops Manager](#) by clicking on the **Logs** tab in the tile and downloading the broker logs.

To access logs using the BOSH CLI, do the following:

1. Identify the on-demand broker (ODB) deployment by running the following command:

```
bosh deployments
```

2. View VMs in the deployment by running the following command:

```
bosh -d DEPLOYMENT-NAME instances
```

3. SSH onto the VM by running the following command:

```
bosh -d service-instance_GUID ssh
```

4. Download the broker logs by running the following command:

```
bosh -d service-instance_GUID logs
```

The archive generated by BOSH or Ops Manager includes the following logs:

Log Name	Description
broker.log	Requests to the on-demand broker and the actions the broker performs while orchestrating the request (e.g. generating a manifest and calling BOSH). Start here when troubleshooting.
broker_ctl.log	Control script logs for starting and stopping the on-demand broker.
post-start.stderr.log	Errors that occur during post-start verification.
post-start.stdout.log	Post-start verification.
drain.stderr.log	Errors that occur while running the drain script.

Access Service Instance Logs and VMs

1. To target an individual service instance deployment, retrieve the GUID of your service instance with the following cf CLI command:

```
cf service MY-SERVICE --guid
```

2. To view VMs in the deployment, run the following command:

```
bosh -d DEPLOYMENT-NAME instances
```

3. To SSH into a VM, run the following command:

```
bosh -d service-instance_GUID ssh
```

4. To download the instance logs, run the following command:

```
bosh -d service-instance_GUID logs
```

Run Service Broker Errands to Manage Brokers and Instances

From the BOSH CLI, you can run service broker errands that manage the service brokers and perform mass operations on the service instances that the brokers created. These service broker errands include:

- `register-broker` registers a broker with the Cloud Controller and lists it in the Marketplace.
- `deregister-broker` deregisters a broker with the Cloud Controller and removes it from the Marketplace.
- `upgrade-all-service-instances` upgrades existing instances of a service to its latest installed version.
- `delete-all-service-instances` deletes all instances of service.
- `orphan-deployments` detects “orphan” instances that are running on BOSH but not registered with the Cloud Controller.

To run an errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand ERRAND-NAME
```

For example:

```
bosh -d my-deployment run-errand deregister-broker
```

Register Broker

The `register-broker` errand registers the broker with Cloud Foundry and enables access to plans in the service catalog. Run this errand whenever the broker is re-deployed with new catalog metadata to update the Cloud Foundry catalog.

Plans with disabled service access are not visible to non-admin Cloud Foundry users, including Org Managers and Space Managers. Admin Cloud Foundry users can see all plans including those with disabled service access.

The errand does the following:

- Registers the service broker with Cloud Controller.
- Enables service access for any plans that have the radio button set to `enabled` in the tile plan page.
- Disables service access for any plans that have the radio button set to `disabled` in the tile plan page.
- Does nothing for any for any plans that have the radio button set to `manual`.

To run the errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand register-broker
```

Deregister Broker

This errand deregisters a broker from Cloud Foundry.

The errand does the following:

- Deletes the service broker from Cloud Controller
- Fails if there are any service instances, with or without bindings

Use the [Delete All Service Instances errand](#) to delete any existing service instances.

To run the errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand deregister-broker
```

Upgrade All Service Instances

If you have made changes to the plan definition or uploaded a new tile into Ops Manager, you might want to upgrade all the MySQL for PCF service instances to the latest software or plan definition.

The `upgrade-all-service-instances` errand does the following:

- Collects all of the service instances the on-demand broker has registered
- For each instance the errand does the following serially
 - Issues an upgrade command to the on-demand broker
 - Regenerates the service instance manifest based on its latest configuration from the tile
 - Deploys the new manifest for the service instance
 - Waits for this operation to complete, then proceeds to the next instance
- Adds to a retry list any instances that have ongoing BOSH tasks at the time of upgrade
- Retries any instances in the retry list until all are upgraded

If any instance fails to upgrade, the errand fails immediately. This prevents systemic problems from spreading to the rest of your service instances.

To run the errand, do one of the following:

- Select the errand through the Ops Manager UI and have it run when you click **Apply Changes**.
- Run the following command.

```
bosh -d DEPLOYMENT-NAME run-errand upgrade-all-service-instances
```

Delete All Service Instances

This errand uses the Cloud Controller API to delete all instances of your broker's service offering in every Cloud Foundry org and space. It only deletes instances the Cloud Controller knows about. It does not delete orphan BOSH deployments.



Note: Orphan BOSH deployments do not correspond to a known service instance. While rare, orphan deployments can occur. Use the `orphan-deployments` errand to identify them.

The `delete-all-service-instances` errand does the following:

1. Unbinds all apps from the service instances.
2. Deletes all service instances sequentially. Each service instance deletion includes:
 - a. Running any pre-delete errands
 - b. Deleting the BOSH deployment of the service instance

- c. Removing any ODB-managed secrets from Credhub
 - d. Checking for instance deletion failure, which results in the errand failing immediately
3. Determines whether any instances have been created while the errand was running. If new instances are detected, the errand returns an error. In this case, Pivotal recommends running the errand again.

⚠ warning: Use extreme caution when running this errand. You should only use it when you want to totally destroy all of the on-demand service instances in an environment.

To run the errand, run the following command:

```
bosh -d service-instance_GUID delete-deployment
```

Detect Orphaned Service Instances

A service instance is defined as “orphaned” when the BOSH deployment for the instance is still running, but the service is no longer registered in Cloud Foundry.

The `orphan-deployments` errand collates a list of service deployments that have no matching service instances in Cloud Foundry and return the list to the operator. It is then up to the operator to remove the orphaned BOSH deployments.

To run the errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand orphan-deployments
```

If orphan deployments exist—The errand script does the following:

- Exit with exit code 10
- Output a list of deployment names under a `[stdout]` header
- Provide a detailed error message under a `[stderr]` header

For example:

```
[stdout]
[{"deployment_name":"service-instance_80e3c5a7-80be-49f0-8512-44840f3c4d1b"}]

[stderr]
Orphan BOSH deployments detected with no corresponding service instance in Cloud Foundry. Before deleting any deployment it is recommended to verify the service instance no longer exists.

Errand 'orphan-deployments' completed with error (exit code 10)
```

These details will also be available through the BOSH `/tasks/` API endpoint for use in scripting:

```
$ curl 'https://bosh-user:bosh-password@bosh-url:25555/tasks/task-id/output?type=result' | jq .
{
  "exit_code": 10,
  "stdout": "[{"deployment_name":"service-instance_80e3c5a7-80be-49f0-8512-44840f3c4d1b"}]\n",
  "stderr": "Orphan BOSH deployments detected with no corresponding service instance in Cloud Foundry. Before deleting any deployment it is recommended to verify the service instance no longer exists.",
  "logs": {
    "blobstore_id": "d830c4bf-8086-4bc2-8c1d-54d3a3c6d88d"
  }
}
```

If no orphan deployments exist—The errand script does the following:

- Exit with exit code 0
- Stdout will be an empty list of deployments
- Stderr will be `None`

```
[stdout]
[]


[stderr]
None

Errand 'orphan-deployments' completed successfully (exit code 0)
```

If the errand encounters an error during running—The errand script does the following:


- Exit with exit 1
- Stdout will be empty
- Any error messages will be under stderr

To clean up orphaned instances, run the following command on each instance:

 **warning:** Running this command may leave IaaS resources in an unusable state.

```
bosh delete-deployment service-instance_SERVICE-INSTANCE-GUID
```

Retrieve Admin and Read-Only Admin Credentials for a Service Instance

1. [Identify the service deployment by GUID.](#)
2. [Log in to BOSH](#) .
3. Open the manifest in a text editor.
4. Look in the manifest for the credentials.

Reinstall a Tile

To reinstall the MySQL for PCF v2.x tile, see the [Reinstalling MySQL for Pivotal Cloud Foundry version 2 and above](#)  Knowledge Base article.

View Resource Saturation and Scaling

To view usage statistics for any service, do the following:

1. Run the following command:

```
bosh -d DEPLOYMENT-NAME vms --vitals
```

2. To view process-level information, run:

```
bosh -d DEPLOYMENT-NAME instances --ps
```


Identify Service Instance Owner


If you want to identify which apps are using a specific service instance from the BOSH deployments name, you can run the following steps:

1. Take the deployment name and strip the `service-instance_` leaving you with the GUID.
2. Log in to CF as an admin.
3. Obtain a list of all service bindings by running the following:

```
cf curl /v2/service_instances/GUID/service_bindings
```

4. The output from the above curl gives you a list of `resources`, with each item referencing a service binding, which contains the `APP-URL`. To find the name, org, and space for the app, run the following:


- a. `cf curl APP-URL` and record the app name under `entity.name`.
- b. `cf curl SPACE-URL` to obtain the space, using the `entity.space_url` from the above curl. Record the space name under `entity.name`.
- c. `cf curl ORGANIZATION-URL` to obtain the org, using the `entity.organization_url` from the above curl. Record the organization name under `entity.name`.

 **Note:** When running `cf curl` ensure that you query all pages, because the responses are limited to a certain number of bindings per page. The default is 50. To find the next page curl the value under `next_url`.

Monitor Quota Saturation and Service Instance Count

Quota saturation and total number of service instances are available through ODB metrics emitted to Loggregator. The metric names are shown below:

Metric Name	Description
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/quota_remaining</code>	global quota remaining for all instances across all plans
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/quota_remaining</code>	quota remaining for a particular plan
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/total_instances</code>	total instances created across all plans
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/total_instances</code>	total instances created for a given plan

 **Note:** Quota metrics are not emitted if no quota has been set.

Knowledge Base (Community)

Find the answer to your question and browse product discussions and solutions by searching the [Pivotal Knowledge Base](#).

File a Support Ticket

You can file a ticket with [Pivotal Support](#). Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

To expedite troubleshooting, provide your service broker logs and your service instance logs. If your `cf service YOUR-SERVICE-INSTANCE` output includes a `task-id`, provide the BOSH task output.

Using MySQL for PCF v2

This topic provides instructions for developers using the MySQL for Pivotal Cloud Foundry (PCF) v2 service for their PCF apps. MySQL provides a relational database for apps and devices.

These procedures use the Cloud Foundry Command Line Interface (cf CLI). For more information, see [Managing Service Instances with the cf CLI](#).

You can also use [Apps Manager](#) to do the same tasks using a graphical UI.

Prerequisites

To use MySQL for PCF v2 with your PCF apps, you need:

- A PCF installation with [MySQL for PCF](#) installed and listed in the [Marketplace](#)
- A [Space Developer](#) or Admin account on the PCF installation
- A local machine with the following installed:
 - a browser
 - a shell
 - the [Cloud Foundry Command-Line Interface](#) (cf CLI)
 - the Linux [watch](#) command
- To [log in to](#) the org and space containing your app

The Process for Using MySQL for PCF in Your App

To use MySQL in a PCF app:

1. Check the service availability in the Marketplace, and see if there is an existing instance of MySQL for PCF in your space.
See [Confirm the MySQL for PCF v2 Service Availability](#), below.
2. If there is no existing instance or you want to use a different one, create an instance of the MySQL for PCF service in the same space as the app.
See [Create a Service Instance](#), below.
3. Push your app into the same space as the MySQL for PCF service instance, using `cf push`.
For information about `cf push`, see [Push](#).
4. Bind the app to the MySQL for PCF service instance, to enable the app to use MySQL.
See [Bind a Service Instance to Your App](#), below.
5. Call the MySQL for PCF service in your app code, and then re-push your app into the space.
See [Use the MySQL Service in Your App](#), below.

Confirm the MySQL for PCF v2 Service Availability

For an app to use the MySQL for PCF v2 service, both of the following must be true:

- The service must be available in the Marketplace for its space.
- An instance of the service must exist in its space.

You can confirm both of these using the cf CLI as follows.

Check Service Availability in the Marketplace

To find out if a MySQL for PCF v2 service is available in the Marketplace, do the following:

1. Enter `cf marketplace`.
2. If the output lists `p.mysql` in the `service` column, MySQL for PCF v2 is available. If it is not available, ask your operator to install it.

```
$ cf marketplace
Getting services from marketplace in org my-org / space my-space as user@example.com...
OK
service      plans      description
[...]
p.mysql      db-small   Dedicated instances of MySQL service to provide a relational database
[...]
```

Check That an Instance Is Running in the Space

To confirm that a MySQL for PCF v2 instance is running in the space, do the following:

1. Use the [cf CLI](#) or [Apps Manager](#) to log in to the org and space that contains the app.
2. Enter `cf services`.
3. Any `p.mysql` listings in the `service` column are service instances of MySQL for PCF v2 in the space.

```
$ cf services
Getting services in org my-org / space my-space as user@example.com...
OK
name      service  plan    bound apps  last operation
my-instance p.mysql  db-small      create succeeded
```

You can [bind](#) your app to an existing instance or [create](#) a new instance to bind to your app.

Create a Service Instance

Unlike pre-provisioned services, on-demand services are created asynchronously, not immediately. The `watch` command shows you when your service instance is ready to bind and use.

To create an instance of the MySQL for PCF v2 service, do the following:

1. Run the command `cf create-service p.mysql PLAN SERVICE-INSTANCE`

Where:

- `PLAN` is the name of the MySQL for PCF v2 plan you want to use.
- `SERVICE-INSTANCE` is a name you choose to identify the service instance. This name appears under `service` in output from `cf services`.

2. Enter `watch cf services` and wait for the `last operation` for your instance to show as `create succeeded`.

```
$ cf create-service p.mysql db-small my-instance

Creating service my-instance in org my-org / space my-space as user@example.com...
OK

$ watch cf services

Getting services in org my-org / space my-space as user@example.com...
OK
name      service  plan    bound apps  last operation
my-instance p.mysql  db-small      create succeeded
```

If you get an error, see [Troubleshooting Instances](#).

Bind a Service Instance to Your App

For an app to use a service, you must bind the app to a service instance. Do this after you push or re-push the app using `cf push`.

To bind an app to a MySQL for PCF instance run the command `cf bind-service APP SERVICE-INSTANCE`

Where:

`APP` is the app you want to use the MySQL service instance.

`SERVICE-INSTANCE` is the name you supplied when you ran `cf create-service`.

```
$ cf bind-service my-app my-instance
```

```
Binding service mydb to my-app in org my-org / space test as user@example.com...
```

```
OK
```

```
TIP: Use 'cf push' to ensure your env variable changes take effect
```

Use the MySQL Service in Your App

To access the MySQL service from your app:

1. Run `cf env APP-NAME` with the name of the app bound to the MySQL for PCF instance.
2. In the output, note the connection strings listed in the `VCAP_SERVICES` > `credentials` object for the app.
3. In your app code, call the MySQL service using the connection strings.

Manage Service Instances

This section describes tasks you do over the life cycle of your apps and data:

- Moving your data to a different plan.
- Removing an app's access to a service it no longer needs.
- Deleting a service instance that is not used.

Update a Service Instance to a Larger Plan

As apps and their databases grow, it may be necessary to update the service instance to a larger plan. This does not require a rebinding of your app. However, while the instance is being migrated to a new service instance, the database will be unavailable for several minutes.

To update a service instance to a larger plan, run the command `cf update-service SERVICE-INSTANCE -p PLAN`

Where:

`PLAN` is the plan you want to upgrade the service instance to.

```
$ cf update-service my-instance -p db-large
```

Unbind an App from a Service Instance

To stop an app from using a service it no longer needs, run the following command to unbind the app from the service:

```
cf unbind-service APP SERVICE-
INSTANCE
```

Where:

`APP` is the app you want to stop using the MySQL service instance.

`SERVICE-INSTANCE` is the name you supplied when you ran `cf create-service`.

```
$ cf unbind-service my-app my-instance
```

```
Unbinding app my-app from service my-instance in org my-org / space my-space as user@example.com...
```

```
OK
```

Delete a Service Instance

You cannot delete a service instance that an app is bound to.

To delete a service instance, do the following:

1. Run the command `cf delete-service SERVICE-INSTANCE`

Where:

`SERVICE-INSTANCE` is the name of the service to delete.

```
$ cf delete-service my-instance
```

```
Are you sure you want to delete the service my-instance ? y
Deleting service my-service in org my-org / space my-space as user@example.com...
OK
```

2. Enter `watch cf service SERVICE-INSTANCE` and wait for a `Service instance not found` error indicating that the instance no longer exists.

MySQL for PCF Tools

The following tools let developers access their MySQL for PCF databases.

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.

The screenshot displays the Pivotal MySQLWeb SQL Worksheet interface. At the top, there's a navigation bar with links like Home, Logout, Menu, Schema Objects, Themes, and a user profile. Below this is a toolbar with buttons for Home, Tables (3), Views (0), Indexes (3), Constraints (3), SQL Worksheet (active), View Connections, Health Endpoints, Information, and Database Variables. A section for 'Query Count', 'Elapsed Time', and 'Explain Plan' all show 'No'. Below this is a text area for running SQL queries. The main area shows a successful execution of a SQL script that drops and creates two tables, 'emp' and 'dept', and inserts a row into 'emp'. The results section at the bottom shows a message: 'Note! Ran 22 Statement(s)' followed by six green status bars indicating successful DDL and DML operations.

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.

See the Pivotal MySQLWeb [code repo](#) and [demo video](#), for how to install and run Pivotal MySQLWeb.

cf CLI MySQL Plugin

To connect to your MySQL for PCF databases from a command line, use the 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.

Example App

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

MySQL Server Defaults

This topic provides information about the defaults that MySQL for Pivotal Cloud Foundry (PCF) applies to its Percona Server components. This topic also provides instructions for how to use optional parameters to change certain server defaults.

Server Defaults

The following table lists the MySQL for PCF server defaults.

Name	Variable Name	Default	Notes
Max Connections	<code>max-connections</code>	750 connections per service instance	System processes count towards this limit.
Max Allowed Packet	<code>max-allowed-packet</code>	256 MB	You can change this size in a session variable if necessary.
Table Definition Cache	<code>table-definition-cache</code>	8192	For more information about updating this variable, see the MySQL documentation .
Reverse Name Resolution	<code>skip-name-resolve</code>	ON	This disables reverse DNS lookups, which improves performance. MySQL for PCF uses user credentials, not hostnames, to authenticate access. Therefore, most deployments do not need reverse DNS lookups. To enable reverse name resolution, clear this option.
Skip Symbolic Links	<code>symbolic-links</code>	OFF	MySQL for PCF is configured to prevent the use of symlinks to tables. This recommended security setting prevents users from manipulating files on the server's file system. For more information, see Making MySQL Secure Against Attackers .
MyISAM Recover Options	<code>myisam-recover-options</code>	BACKUP, FORCE	This setting enables MySQL for PCF to recover from most MyISAM problems without human intervention. For more information, see the MySQL documentation .
Log Bin Trust Function Creators	<code>log-bin-trust-function-creators</code>	ON	This setting relaxes constraints on how MySQL writes stored procedures to the binary log. For more information, see the MySQL documentation .
Event Scheduler	<code>event-scheduler</code>	ON	MySQL for PCF enables the event scheduler so users can create and use events in their dedicated service instances.
Lower Case Table Names	<code>lower-case-table-names</code>	0	By default, all table names are case sensitive. Operators can change this default on the MySQL Configuration page , and may allow developers to override the default when creating a service instance. For more information about the use for lowercase table names, see the MySQL documentation .
Audit Log	<code>audit-log</code>	OFF	When enabled on the MySQL Monitoring pane , logs are written as CSVs to <code>/var/vcap/sys/log/mysql/mysql-audit-log</code> as well as a remote syslog drain if it is enabled.

InnoDB Buffer Pool Size	<code>innodb-buffer-pool-size</code>	50% of the available memory on each service instance	Dynamically configured to be 50% of the available memory on each service instance.
InnoDB Log File Size	<code>innodb-log-file-size</code>	256 MB	MySQL for PCF clusters default to a log-file size of 256 MB.
InnoDB Log Buffer Size	<code>innodb-log-buffer-size</code>	32 MB	MySQL for PCF defaults to 32 MB to avoid excessive disk I/O when issuing large transactions.
InnoDB Auto Increment Lock Mode	<code>innodb-autoinc-lock-mode</code>	2	Auto Increment uses “interleaved” mode. This enables multiple statements to execute at the same time. There may be gaps in auto-incrementing columns.
Collation Server	<code>collation-server</code>	<code>utf8-general-ci</code>	You can override this during a session.
Character Set	<code>character-set-server</code>	<code>utf8</code>	Defaults all character sets. You can override this during a session.

Changing MySQL Server Defaults

You can set optional parameters to change the MySQL for PCF server defaults to accommodate apps with a read-heavy or write-heavy workload, or if you need to use lowercase table names.

The procedures in this section use the Cloud Foundry Command Line Interface ([cf CLI](#)). You can also use [Apps Manager](#) to perform the same tasks using a graphical UI.

For general information about using MySQL for PCF, see [Using MySQL for PCF](#).

Optional Parameters for the MySQL for PCF Service Instances

MySQL for PCF service instances are configured by default with industry best practices. For more specific use cases, you can customize the following parameters:

Key	Type	Default	Description
<code>workload</code>	String	<code>mixed</code>	You can set this to <code>read-heavy</code> , <code>mixed</code> , or <code>write-heavy</code> . For more information, see Understanding Workload Types .
<code>enable_lower_case_table_names</code>	Boolean	Set by the operator	The operator sets a default for this parameter, and may allow you to override the default. If allowed, you can only set this when creating a service instance. If you set this to <code>true</code> , table names are stored in lowercase. For more information, see Understanding Lowercase Table Names .

Use Optional Parameters

You can change these default configuration parameters using the cf CLI as follows:

- `workload` —When creating a new instance, or updating an existing one
- `enable_lower_case_table_names` —When creating a new instance

Change Default Parameters when Creating a Service Instance

To create a service instance using optional parameters use the following command:

```
cf create-service SERVICE-INSTANCE -c '{ "PARAMETER": "PARAMETER-VALUE"
}'
```

The `-c` flag accepts a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file.

For example:

```
$ cf create-service myDB -c '{ "workload": "mixed" }'
Creating service instance myDB in org system / space system as admin...
OK

Create in progress. Use 'cf services' or 'cf service myDB' to check operation status.
```

If you get an error using optional parameters, see [Troubleshooting Instances](#).

Change Default Parameters on an Existing Service Instance

To change the default parameters of an existing instance use the following command:

```
cf update-service SERVICE-INSTANCE -c '{ "PARAMETER": "PARAMETER-VALUE"
}'
```

The `-c` flag accepts a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file.

For example:

```
$ cf update-service myDB -c '{ "workload": "mixed" }'
```

If you get an error using optional parameters, see [Troubleshooting Instances](#).

Understanding Workload Types

MySQL for PCF offers three workload profiles that developers can use to configure MySQL instances based on their specific app workloads. For instructions on changing configuration options, see [Use Optional Parameters](#).

Profile	Description
Mixed Workload	By default, each MySQL service instance is configured for a mixed workload, that is, a workload that is equally heavy on reads and writes. The configuration for this profile is described in detail in MySQL Server Defaults .
Read-Heavy Workload	For apps that have a large number of reads, the service instance can be configured for a read-heavy workload. The read-heavy profile changes these two properties from the MySQL Server Defaults : <ul style="list-style-type: none"> <code>innodb_buffer_pool_size</code> is increased to <code>75%</code> of the MySQL VM's memory <code>innodb_flush_method</code> is set to <code>O_DIRECT</code>
Write-Heavy Workload	For apps that write to the database a lot, the service instance can be configured for a write-heavy workload. The write-heavy profile changes these four properties from the MySQL Server Defaults : <ul style="list-style-type: none"> <code>innodb_buffer_pool_size</code> is increased to <code>75%</code> of the MySQL VM's memory <code>innodb_flush_method</code> is set to <code>O_DIRECT</code> <code>innodb_log_file_size</code> is increased to <code>1GB</code> <code>max_allowed_packets</code> is increased to <code>1GB</code>

Understanding Lowercase Table Names

If you are migrating a database from a system that was case insensitive, that is, where `TableName` is the same as `TABLEName`, then forcing lowercase turns both into `tablename`. In this way both names are interpreted as the same table.

For more information, see the MySQL [documentation](#).

Customizing Database Credentials

This topic provides instructions for developers to customize access credentials and privileges for MySQL for Pivotal Cloud Foundry (PCF) service instances.

These procedures use the Cloud Foundry Command Line Interface (cf CLI). You can also use [Apps Manager](#) to do the same tasks using a graphical UI.

For information about using MySQL for PCF, see [Using MySQL for PCF](#).

Create Read-only Access Credentials

MySQL for PCF lets space developers create read-only credentials to give to users who need read-only access to the database. You can create credentials by using the cf CLI to create a new [service key](#).

To create and find read-only credentials for an existing service instance, do the following:

1. Run the following command to create a new service key for a service instance, specifying it to be read-only:

```
cf create-service-key SERVICE-INSTANCE KEY-NAME -c '{ "read-only": true }'
```

Where:

SERVICE-INSTANCE: The name of the service instance

KEY-NAME: What you want to call the key

For example,

```
$ cf create-service-key mydb mykey1 -c '{ "read-only": true }'
Creating service key mykey1 for service instance mydb as admin...
OK
```


2. Run the following command to retrieve the read-only credentials from the service key:

```
cf service-key SERVICE-INSTANCE KEY-NAME
```

For example,

```
$ cf service-key mydb mykey1

{
  "hostname": "99.99.99.9",
  "jdbcUrl": "jdbc:mysql://99.99.99.9:3306/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@99.99.99.9:3306/cf_e2d148a8_1baa_4961_b314_2431f57037e5?reconnect=true",
  "username": "abcdefghijklm"
}
```

 **Note:** Any user with access to create a service key may provision a fully privileged service key.

Create Credentials for a Custom Username

If you have users that want to access the database with a specific username, MySQL for PCF lets space developers create custom usernames for service keys or service bindings. You can create credentials by using the cf CLI to create a new [service key](#).

To create and find custom username credentials for an existing service instance, do the following:

1. Create a new service key for a service-instance, specifying a username:

```
$ cf create-service-key SERVICE-INSTANCE KEY-NAME -c '{ "username": NEW-USER-NAME }'
```

For example,

```
$ cf create-service-key mydb mykey2 -c '{ "username": myuser }'
Creating service key mykey2 for service instance mydb as admin...
OK
```

2. Run the following command to retrieve the credentials from the service key.

```
cf service-key SERVICE-INSTANCE KEY-NAME
```

For example,

```
$ cf service-key mydb mykey2
```

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



Note: Any user with access to create a service key may provision a fully privileged service key.

Migrating Data from MySQL for PCF v1 to v2


This topic describes how to migrate the data from a MySQL for Pivotal Cloud Foundry (PCF) v1.x service instance to a MySQL for PCF v2.x service instance.

Prerequisites

To perform the procedures in this topic, you must have the following:

- An existing MySQL for PCF v1.x service instance with the data you want to migrate
- MySQL for PCF v2.x installed in the same PCF environment as your PCF v1.x service instance
- At least one MySQL for PCF v2.x service plan available in the same space as your MySQL for PCF v1.x service instance. The persistent disk size of this service plan must be three times larger than the maximum size of the data you store in it. For more information, see [MySQL Persistent Disk Usage](#).

 **Note:** To view available service plans, run `cf marketplace`. MySQL for PCF v2.x appears as `p.mysql` and MySQL for PCF v1.x appears as `p-mysql`.

 **Warning:** Migrating large datasets can take several hours. Migration of the data is linear, and depends on the hardware being used. For example, if X amount of data takes 10 minutes to migrate, then 2X amount of data will take 20 minutes to migrate using the same hardware. Do a test migration with small datasets to estimate how long the entire migration will take before migrating larger datasets.

Stop and Unbind Apps

Do the following to stop and unbind any apps that use your MySQL for PCF v1.x service instance:

1. Use the cf CLI to target and log in to your PCF deployment. For example:

```
$ cf target api.example.com
$ cf login
```

When prompted, enter your credentials and target the org and space where the MySQL for PCF v1.x is located.

2. Stop any traffic to the MySQL for PCF v1.x database by stopping any apps that are bound to the service instance. Run the following command for each app bound to the instance:

```
cf stop APP
```

Where `APP` is the name of your app.

For example:

```
$ cf stop my-app
Stopping app my-app in org my-org / space my-space as user@example.com...
OK
```

To retrieve a list of bound apps, run `cf services`.

3. Unbind your app from the MySQL for PCF v1.x service instance. Run the following command for each app bound to the instance:

```
cf unbind-service APP V1-INSTANCE
```

Where:

- `APP` is the name of your app.
- `V1-INSTANCE` is the name of your MySQL for PCF v1.x service instance.

For example:

```
$ cf unbind-service my-app my-instance
Unbinding app my-app from service my-v1-instance in org my-org / space my-space as user@example.com...
OK
```

Migrate Data

To migrate the data from the MySQL for PCF v1.x service instance to a new MySQL for PCF v2.x service instance, use the `cf mysql-tools plugin`.

The `migrate` command of the plugin renames the MySQL for PCF v1.x service instance by appending `-old` and gives the original name of the instance to the newly created MySQL for PCF v2.x service instance.

⚠ warning: The `migrate` command does not migrate all stored code. The command only migrates views and does not migrate triggers, routines, or events. To manually migrate triggers, routines, or events, contact [Pivotal Support](#).

Do the following to migrate data to an instance of the same name:

1. Install the `cf mysql-tools` plugin. Run the following command:

```
cf install-plugin -r CF-Community "MysqlTools"
```

2. Use the plugin to migrate your data. Run the following command:

```
cf mysql-tools migrate V1-INSTANCE V2-PLAN
```

Where:

- `V1-INSTANCE` is the name of your MySQL for PCF v1.x service instance.
- `V2-PLAN` is the name of the MySQL for PCF v2.x service plan to use for the new MySQL for PCF v2.x service instance. For example, `db-small`. To view available MySQL for PCF v2.x service plans, run `cf marketplace` and locate the plans under `p.mysql`.

For debug purposes, optionally add the `--no-cleanup` flag. This flag preserves the app that runs the migration task and the newly created service instances if the migration is not successful. However, if a migration succeeds, the migration app is cleaned even if the `--no-cleanup` flag is provided.

💡 Note: An error message appears if the migration is not successful. If the migration is successful, a `Migration completed successfully` message appears in the messages returned to you after you run the migration command, as show in the example below. This message is the best indication that your migration process was successfully completed.

For example:

```
$ cf mysql-tools migrate my-instance db-small
2018/04/24 11:31:19 Creating new service instance "my-instance" for service p.mysql using plan db-small
2018/04/24 11:41:01 Unpacking assets for migration to /var/folders/dm/66n2j9xx02l8vs58q2whz4080000gn/T/migrate_app_101341527
2018/04/24 11:41:02 Started to push app
Done uploading
2018/04/24 11:41:09 Successfully pushed app
2018/04/24 11:41:10 Successfully bound app to v1 instance
2018/04/24 11:41:12 Successfully bound app to v2 instance
2018/04/24 11:41:12 Starting migration app
2018/04/24 11:41:25 Started to run migration task
2018/04/24 11:41:27 Migration completed successfully
2018/04/24 11:41:29 Cleaning up...
```

Developers can SSH into the app container to examine the logs.

Validate Data

After migrating your data, you must verify that the data has successfully migrated by validating the data in the new MySQL for PCF v2.x service instance. You can validate the data by creating an SSH tunnel to gain direct command line access to the new MySQL for PCF v2.x service instance.

Do the following to create an SSH tunnel to the instance and validate your data:

1. Perform the steps in [Push Your Host App](#) to push an app that will act as the host for the SSH tunnel.
2. Perform the steps in [Create Your Service Key](#) to create a service key for your MySQL for PCF v2.x service instance.
3. Perform the steps in [Configure Your SSH Tunnel](#) to configure an SSH tunnel to your MySQL for PCF v2.x service instance.
4. Perform the steps in [Access Your Service Instance](#) to establish direct command-line access to your MySQL for PCF v2.x service instance.
5. From the MySQL shell, validate that the data that you expect to see has been imported into the MySQL for PCF v2.x service instance.
6. Exit the MySQL shell and kill the SSH tunnel.

Rebind and Restage Apps

To complete the migration, rebind and restage any apps that had been bound to the original MySQL for PCF v1.x service instance.

Do the following to rebind and restage your apps:

1. Bind the app to the new service instance. Run the following command:

```
cf bind-service APP V2-INSTANCE
```

Where:

- `APP` is the name of your app.
- `V2-INSTANCE` is the name of your MySQL for PCF v2.x service instance.

For example:

```
$ cf bind-service my-app my-instance-new
Binding service my-instance-new to app my-app in org my-org / space my-space as user@example.com...
OK
TIP: Use 'cf restage my-app' to ensure your env variable changes take effect
```

2. Restage the app. For example:

```
$ cf restage my-app
Restaging app my-app in org my-org / space my-space as user@example.com...
[...]
```

The app is now using your new MySQL for PCF v2.x service instance and should be operational again.

Delete the Old Database

After rebinding and restaging your apps to confirm that migration was successful, Pivotal recommends saving resources by deleting the old database instance.

To perform the deletion, run the following command:

```
cf delete-service SERVICE_INSTANCE
```

Where `SERVICE_INSTANCE` is the name of your old database instance.

For example:

```
$ cf delete-service my-instance
```

MySQL Environment Variables

This topic provides a reference for the environment variables that Cloud Foundry stores for MySQL for PCF service instances. These variables include the credentials that apps use to access the service instances.

VCAP_SERVICES

Applications running in Cloud Foundry gain access to the bound service instances via an environment variable credentials hash called `VCAP_SERVICES`. An example hash is show below:

```
{
  "p.mysql": [{
    "label": "p.mysql",
    "name": "my-instance",
    "plan": "db-medium",
    "provider": null,
    "syslog_drain_url": null,
    "tags": [
      "mysql"
    ],
    "credentials": {
      "hostname": "10.0.0.20",
      "jdbcUrl": "jdbc:mysql://10.0.0.20:3306/service_instance_db?user=fefcbe8360854a18a7994b870e7b0bf5\u0026password=z9z6eskdbslrhtxt",
      "name": "service_instance_db",
      "password": "z9z6eskdbslrhtxt",
      "port": 3306,
      "uri": "mysql://fefcbe8360854a18a7994b870e7b0bf5:z9z6eskdbslrhtxt@10.0.0.20:3306/service_instance_db?reconnect=true",
      "username": "fefcbe8360854a18a7994b870e7b0bf5"
    },
    "volume_mounts": []
  }]
}
```

You can search for your service by its `name`, given when creating the service instance, or dynamically via the `tags` or `label` properties. The `credentials` property can be used as follows:

- The `credentials` properties `uri`, `name`, `hostname`, `port`, `username`, `password`, and `jdbcUrl` provide access to the MySQL protocol.

In common with all services in [Pivotal Cloud Foundry](#) (PCF), the `VCAP_SERVICES` environment variable for an app is only modified when the app is bound to a service instance. Users will need to `cf unbind-service`, `cf bind-service` and `cf restage` their app in this scenario.

Troubleshooting On-Demand Instances

This topic provides techniques app developers can use to begin troubleshooting on-demand instances.

If Instances or Database are Inaccessible

You might experience the following during upgrades or in a leader-follower topology.

Temporary Outages

MySQL for PCF service instances can become temporarily inaccessible during upgrades and VM or network failures. For more information, see [Service Interruptions](#).

Apps Cannot Write to the Database

If you have a leader-follower topology and your apps can no longer write to the database, the leader VM might be read-only. Contact your operator to check if the leader VM is read-only. For more information about troubleshooting this problem, see [Both Leader and Follower Instances Are Read-Only](#).

If you can no longer read to the database as well, your persistent disk might be full. For more information about troubleshooting inoperable apps, see [Apps are Inoperable](#).

Apps are Inoperable

If your apps become inoperable, your persistent disk might be full. In this state, read, write, and Cloud Foundry Command-Line Interface (cf CLI) operations do not work.

Contact your operator to check if your persistent disk is full. For more information about troubleshooting this problem, see [Persistent Disk is Full](#).

Troubleshoot Errors

You might see an error when using the Cloud Foundry Command-Line Interface (cf CLI) to perform basic operations on a MySQL for PCF service instance:

- `cf create`
- `cf update`
- `cf bind`
- `cf unbind`
- `cf delete`

Understand a Cloud Foundry (CF) Error Message

Failed operations (create, update, bind, unbind, delete) result in an error message. You can retrieve the error message later by running the cf CLI command `cf service INSTANCE-NAME`.

```
$ cf service myservice
```

```
Service instance: myservice
Service: super-db
Bound apps:
Tags:
Plan: dedicated-vm
Description: Dedicated Instance
Documentation url:
Dashboard:
```

```
Last Operation
```

```
Status: create failed
```

```
Message: Instance provisioning failed: There was a problem completing your request.
```

```
Please contact your operations team providing the following information:
```

```
service: redis-acceptance,
```

```
service-instance-guid: ae9e232c-0bd5-4684-af27-1b08b0c70089,
```

```
broker-request-id: 63da3a35-24aa-4183-aec6-db8294506bac,
```

```
task-id: 442,
```

```
operation: create
```

```
Started: 2017-03-13T10:16:55Z
```

```
Updated: 2017-03-13T10:17:58Z
```

Use the information in the `Message` field to debug further. Provide this information to Pivotal Support when filing a ticket.

The `task-id` field maps to the BOSH task ID. For more information on a failed BOSH task, use the `bosh task TASK-ID`.

The `broker-request-guid` maps to the portion of the On-Demand Broker log containing the failed step. Access the broker log through your syslog aggregator, or access BOSH logs for the broker by typing `bosh logs broker 0`. If you have more than one broker instance, repeat this process for each instance.

Find Information about Your Service Instance

You might need to find the name, GUID, or other information about a service instance. To find this information, do the following:

1. Log into the space containing the instance or failed instance.

```
$ cf login
```

2. If you do not know the name of the service instance, run `cf services` to see a listing of all service instances in the space. The service instances are listed in the `name` column.

```
$ cf services
Getting services in org my-org / space my-space as user@example.com...
OK
name      service  plan    bound apps  last operation
my-instance  p.mysql  db-small  create succeeded
```

3. To retrieve more information about a specific instance, run `cf service SERVICE-INSTANCE-NAME`

4. To retrieve the GUID of the instance, run `cf service SERVICE-INSTANCE-NAME --guid`

The GUID is useful for debugging.

Use the Knowledge Base (Community)

Find the answer to your question and browse product discussions and solutions by searching the [Pivotal Knowledge Base](#).

File a Support Ticket

You can file a support ticket [here](#). Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

To expedite troubleshooting, if possible, provide your service broker logs, service instance logs, and BOSH task output. Your cloud operator should be able to obtain these from your error message.

