Pivotal

PRODUCT DOCUMENTATION

MySQL for PCF®

Version 2.1

User's Guide

Table of Contents

MySQL for Pivotal Cloud Foundry

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

Note: This documents MySQL for PCF v2.1, 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 dropdown above.

About MySQL for PCF

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

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 VM that serves a single service instance	PCF creates each VM on-demand when app developer creates service instance	`p.mysql` (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	`p-mysql` (with a dash)

Product Snapshot

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

Element	Details
Version	v2.1.5
Release date	May 8, 2018
Software component version	Percona Server v5.7.20–21
Compatible Ops Manager version(s)	v1.11.x and v1.12.x
Compatible Elastic Runtime version(s)	v1.11.x and v1.12.x
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere

Overview

The MySQL for PCF product delivers dedicated instances on demand, "Database as a Service", to Cloud Foundry users. When installed, the tile deploys and maintains a single Service Broker that is responsible for Cloud Foundry 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 v2.1, see Minor Overview.

MySQL for PCF and Other PCF Services

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

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

The following PCF services offer on-demand service plans:

- MySQL for PCF v2.0 and later

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

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

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

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

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

Feedback

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

Overview of MySQL for PCF v2.1

This topic describes the significant new features in MySQL for PCF v2.1. 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 **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 system 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	Assigning multiple AZs to MySQL jobs does not guarantee high availability.	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. However each instance still operates as a single node with no clustering.

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.

v2.1.5

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.
- Requires stemcell 3445.42.

v2.1.4

Release Date: March 15, 2018

- Upgrades Percona Server to 5.7.20-21.
- Requires stemcell 3445.28.

v2.1.3

Release Date: February 8, 2018

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

v2.1.2

Release Date: January 19, 2018

- Requires stemcell 3445.23.
- Protects against orphaning a disk when updating a service instance to a plan in a different availability zone (AZ).

v2.1.1

Release Date: November 15, 2017

- Requires stemcell 3445.16.

v2.1.0

Release Date: October 15, 2017

- Provides a new restore utility on each service instance to make restoring from a backup artifact easier. For more information, see Restore the Service Instance.
- Adds the ability to enable or disable `lower_case_table_names` for all MySQL service instances or only specific service instances. This helps when migrating from legacy systems that need case insensitivity. For more informatation, see configure MySQL.
- Changes several MySQL server default configurations to provide better consistency and expected behavior when migrating from the MySQL for PCF v1 series.

- Upgrades several dependencies including, `golang 1.9.0` and `xtrabackup 2.4.5`.
- The origin for all MySQL metrics is now `p.mysql` rather than `p-mysql` to better reflect the service name.
- Requires stemcell 3421.20.
- This version of the tile can only be installed in an Ops Manager v1.11 or later environment.
- Fixes a known issue in the v2.0 series where smoke tests failed if port 80 was blocked for internal traffic. All traffic from the smoke-tests errand now goes through port 443.

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.

BOSH 2.0 and the Service Network

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 Ops Manager 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

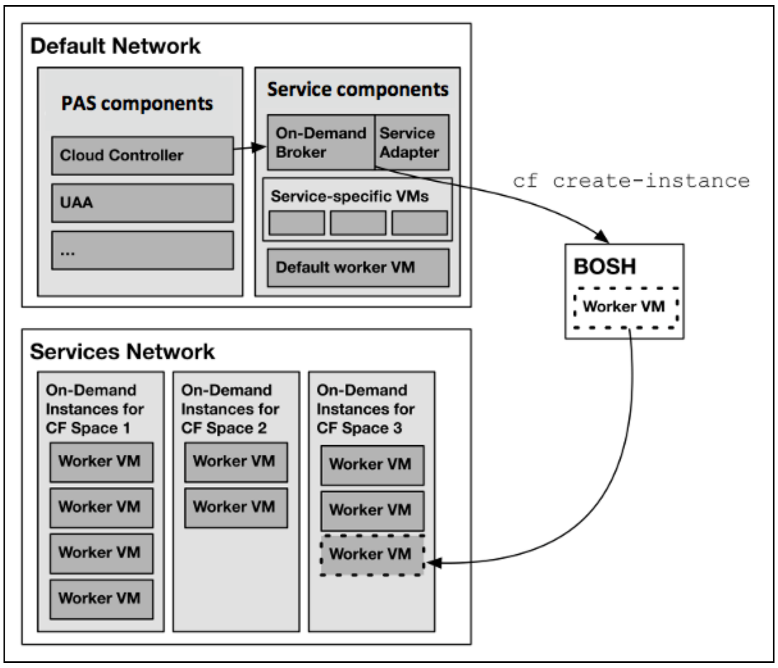
On-demand PCF services rely on the BOSH 2.0 ability to dynamically deploy VMs in a dedicated network. The on-demand service broker uses this capability to create single-tenant service instances in a dedicated service network.

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 application 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

Prior to deploying any service tile that uses the on-demand broker (ODB), the operator must request the network connections needed to allow various components of Pivotal Cloud Foundry (PCF) to communicate with ODB. The specifics of how to open those connections varies for each IaaS.

The following table shows the responsibilities of the key components in an on-demand architecture.

Key Components	Their Responsibility
BOSH Director	Creates and updates service instances as instructed by ODB
BOSH Agent	BOSH includes an Agent on every VM that it deploys. The Agent listens for instructions from the Director and carries out those instructions. The Agent receives job specifications from the Director and uses them to assign a role, or Job, to the VM.
BOSH UAA	As an OAuth2 provider, BOSH UAA issues tokens for clients to use when they act on behalf of BOSH users.
ERT	Contains the apps that are consuming services
ODB	Instructs BOSH to create and updated 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
ODB	• BOSH Director • BOSH UAA	• 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	ERT	8443	One-way	The default port is not configurable.
Errand VMs	• ERT • ODB • Deployed Service Instances	• 8443 • 8080 • Specific to the service. May be one or more ports.	One-way	The default port is not configurable.
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.
Deployed apps on ERT	Deployed service instances	Specific to the service. May be one or more ports.	One-way	This connection is for general use, app-specific tasks. Avoid opening administrative ports for this connection.
ERT	ODB	8080	One-way	This port allows Elastic Runtime to communicate with the ODB component.

MySQL Server Defaults

This section describes the defaults that the MySQL for PCF tile applies to its Percona Server components. There are other components that can be customized as well.

Max Connections

All service instances accept up to 750 connections. System processes count towards this limit.

Max Allowed Packet: 256 MB

MySQL for PCF allows blobs up to 256 MB in size. You can change this size in a session variable if necessary.

Table Definition Cache: 8192

For more information about updating this variable, see the MySQL documentation.

Reverse Name Resolution Off

Disabling reverse DNS lookups improves performance. To enable reverse name resolution, clear this option. A cleared reverse name resolution causes the MySQL servers to perform a reverse DNS lookup on each new connection. Typically, MySQL restricts access by hostname, but MySQL for PCF uses user credentials, not hostnames, to authenticate access. Because of this, most deployments do not need reverse DNS lookups.

Skip Symbolic Links

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.

MyISAM Recover Options: 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: ON

This setting relaxes certain constraints on how MySQL writes stored procedures to the binary log. For more information, see the MySQL documentation.

Event Scheduler: ON

MySQL for PCF enables the event scheduler so users can create and utilize events in their dedicated service instances.

Lower Case Table Names: ON

By default, all table names are case sensitive. This option may be modified on the MySQL Configuration page. For more information about the use for lowercase table names, see the MySQL documentation.

Audit Log: OFF

The MySQL audit log is off by default. When enabled on the MySQL Monitoring page, logs are written as CSVs to `/var/vcap/sys/log/mysql/mysql_audit_log` as well as a remote syslog drain if it is enabled.

InnoDB Buffer Pool Size

Dynamically configured to be 50% of the available memory on each service instance.

InnoDB Log File Size: 256 MB

MySQL for PCF clusters default to a log-file size of 256 MB.

InnoDB Log Buffer Size: 32 MB

MySQL for PCF defaults to 32 MB to avoid excessive disk I/O when issuing large transactions.

InnoDB Auto Increment Lock Mode: 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: UTF8 General CI

MySQL for PCF defaults the collation server to `utf8_general_ci`. You can override this during a session.

Character Set: UTF8

MySQL for PCF defaults all character sets to `utf8`. You can override this during a session.

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.

Service Plan Recommended Usage and Limitations

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

MySQL On-Demand plans use dedicated VM and disks, which will consume IaaS resources. Operators can limit resource usage with Plan Quotas and a Global Quota, but resource usage will vary based on number of On-Demand instances provisioned.

If the number of on-demand instances is greater than or equal to the Global Quota set on the ‘On Demand Service Settings’ page, no new instances can be provisioned.

To calculate the maximum cost/ usage for each plan:

```
max_plan_resources = plan_quota x plan_resources
```

To calculate the maximum cost across plans, add together the cost/ usage for each plan, while the quotas sum to less than the global quota.

```
While (plan_1_quota + plan_2_quota) ≤ global_quota:
max_resources = (plan_1_quota x plan_1_resources) + ( plan_2_quota x plan_2_resources)
```

To calculate the current IaaS cost/ usage across On-Demand plans:

1. Current instances provisioned for all plans can be found by referencing the `total_instance` metric.
2. Multiple the `total_instance` for each plan by that plan’s resources. Sum all plans that are active to get your total current usage

```
current_usage = (plan_1_total_instances x plan_1_resources) + (plan_2_total_instances x plan_2_resources)
```

Availability Using Multiple AZs

MySQL for PCF v2 supports configuring multiple availability zones but this configuration does not provide high availability.

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 Utilization

MySQL backups are taken on the VM and temporarily stored and encrypted on the VM. This results in increased disk utilization while backups are being taken. When enabling backups, we recommend choosing persistent disk sizes that are three times larger than you intend to be available to app developers using the service.

Backups require two to three times as much disk space as the server needs. The first is for the actual data, the second is the copied backup, and the third is for the encrypted backup.

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 v2.1 tile. The MySQL for PCF v2.1 service lets PCF developers create and use their own MySQL service instances on demand.

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

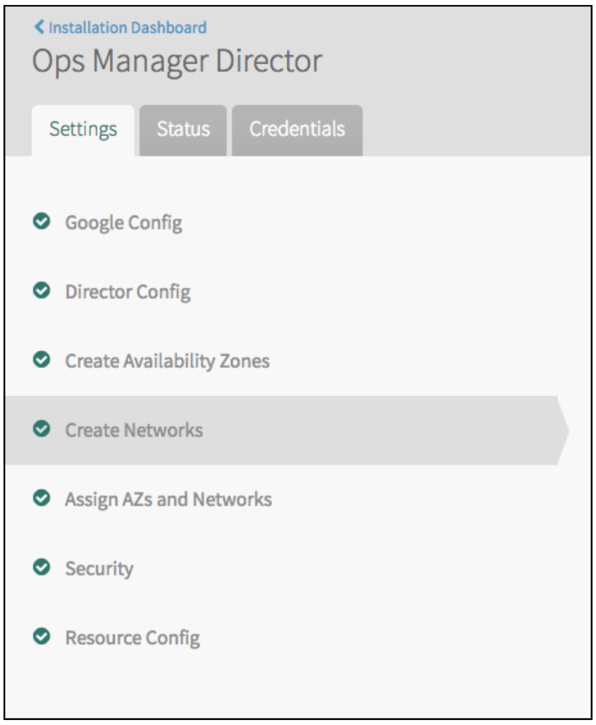
To let apps running on Cloud Foundry communicate with the MySQL service network, you must create an Application Security Group (ASG).

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.

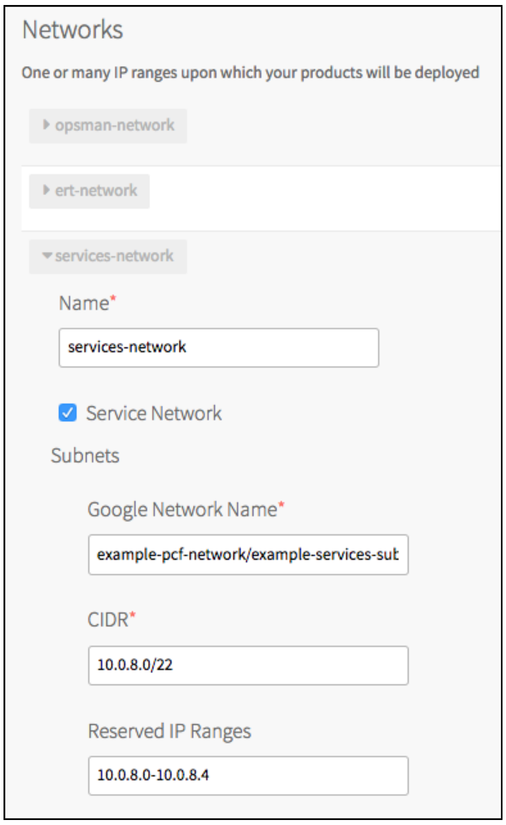
The example below follows this procedure to create an ASG.

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-sub

CIDR*

10.0.8.0/22

Reserved IP Ranges

10.0.8.0-10.0.8.4

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"
  }
]
```

5. 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
```

6. 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:

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

For general information about the Ops Manager Resurrector, see Using Ops Manager Resurrector

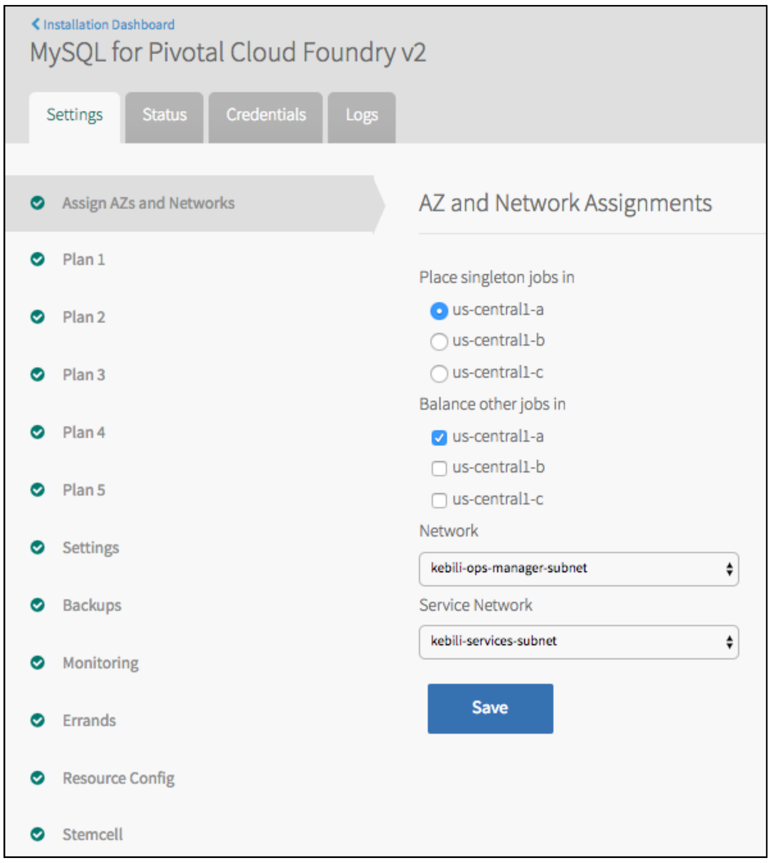
Download and Install the Tile

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.



AZ and Network Assignments

Place singleton jobs in
(●) us-central1-a
( ) us-central1-b
( ) us-central1-c

Balance other jobs in
[✓] us-central1-a
[ ] us-central1-b
[ ] us-central1-c

Network: kebili-ops-manager-subnet

Service Network: kebili-services-subnet

Configure the Tile

Follow the steps below to configure the MySQL for PCF service. MySQL for PCF v2.1 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 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**.

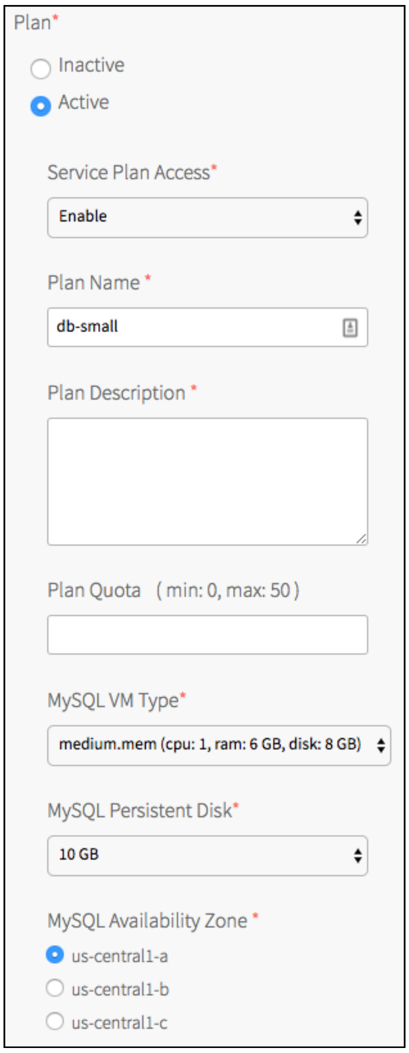
Configure Service Plans

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]

Service Plan Access*: Enable

Plan Name*: db-small

Plan Description*:

Plan Quota (min: 0, max: 50):

MySQL VM Type*: medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)

MySQL Persistent Disk*: 10 GB

MySQL Availability Zone*
[✓ us-central1-a] us-central1-b us-central1-c

3. Configure the fields as follows:

Field	Instructions
Service Plan Access	Select one of the following options: ◦ **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 PCF 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.
MySQL Persistent Disk	Select a disk size. This disk stores the MySQL messages. **Note**: If you intend to enable backups, choose a persistent disk type that is three times as large as you intend to provide to developers. For more information, see Backing Up and Restoring On-Demand MySQL.
MySQL Availability Zone	Select an AZ for the node. This AZ should be different than the AZ that the broker is in. The plan creates all service instance VMs in this AZ. To prevent orphaning a disk, only migrate service instances to a different plan if the plan is in the same AZ.

4. Click **Save**.

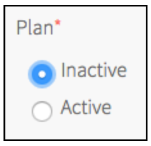
WARNING: If you expect your developers to upgrade from one plan to another, do not place the plans in separate AZs. For example, if you create Plan 1 in AZ1 and Plan 2 in AZ2, developers receive an error and cannot continue if they try to upgrade from Plan 1 to Plan 2. This prevents them from losing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

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

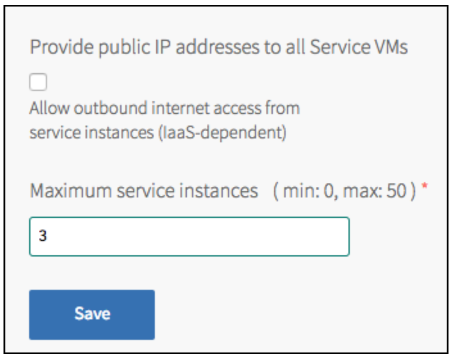

Plan*
[✓ Inactive] Active

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.

1. 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

2. Configure the fields as follows:

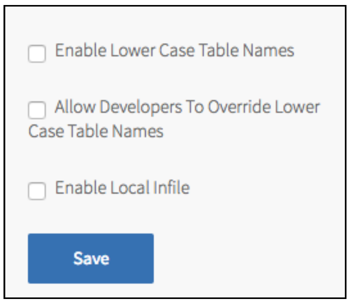
Field	Instructions
Provide public IP addresses to all Service VMs	Select this checkbox: ◦ If the service instances need an external backup, blobstore, or syslog storage ◦ If you have configured BOSH to use an external blobstore.
Maximum service instances	Enter the global quota for all on-demand instances 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

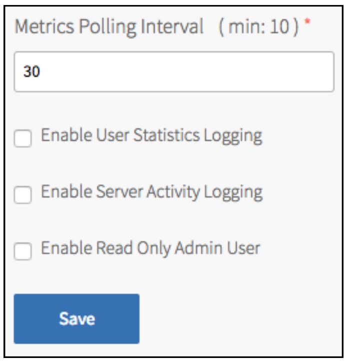
2. Configure the fields as follows:

Field	Instructions
Enable Lower Case Table Names	Select this checkbox to set `lower_case_table_names` to 1 on all MySQL for PCF instances. For more information read the MySQL documentation.
Allow Developers To Override Lower Case Table Names	Select this checkbox to allow developers to override the default Lower Case Table Names value set above. By default, all service instances will be created with the value based on "Enable Lower Case Table Names" above. Checking this box allows developers to modify it using the cf CLI.
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 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

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	Select this checkbox to create a read-only admin user, `roadmin`, on each service instance. This user can be used for auditing and monitoring without risking mutating or changing any data, because the `roadmin` user cannot make changes. To retrieve the credentials for the `roadmin` user, see Retrieve Admin and Read-Only Admin Credentials for a Service Instance. The read-only admin user is always `roadmin`, but the password varies by service instance.

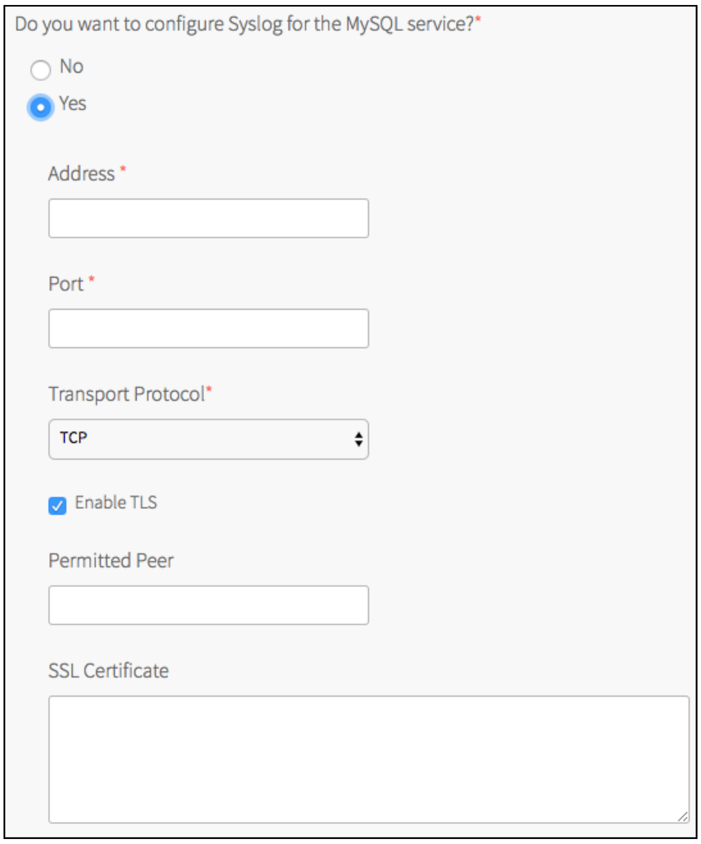
3. Click **Save**.

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, `logmanager.example.com`.
Port	Enter the port of the syslog server for sending logs, for example, `29279`.
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, `*.example.com`
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.

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.

Enabling Automated Backups

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

- **How and where**—There are four options for how automated backups transfer backup data and where the data saves 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 tiles 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 enable and 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.

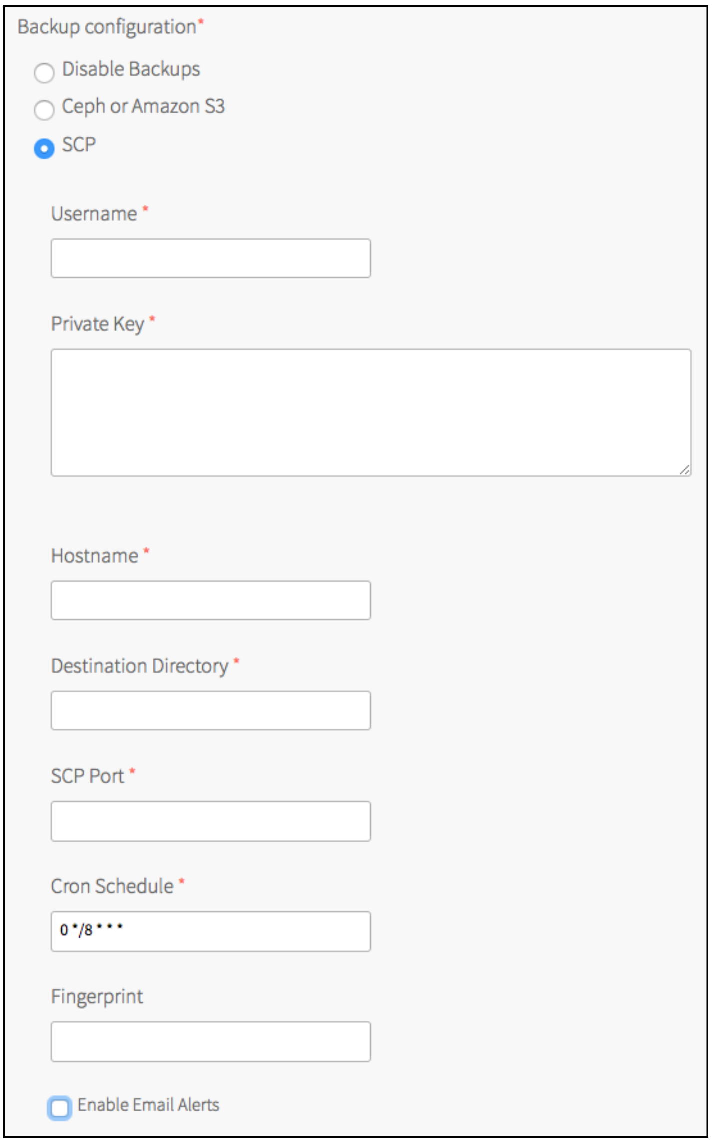
   Note: Pivotal recommends using a VM outside the PCF deployment for the destination of SCP backups. As a result you may need to enable public IPs for the MySQL VMs.

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*

Disable Backups [ ]
Ceph or Amazon S3 [ ]
[✓ SCP]

Username*

Private Key*

Hostname*

Destination Directory*

SCP Port*

Cron Schedule*

0 */8 * * *

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 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.

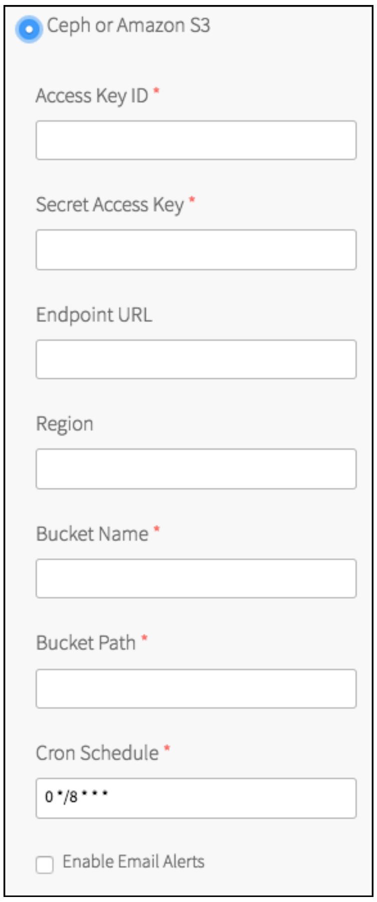
3. Attach the policy you created to the AWS user account that MySQL for PCF will use to access S3 (**IAM** > **Policies** > **Policy Actions** > **Attach**).

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**.



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 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.

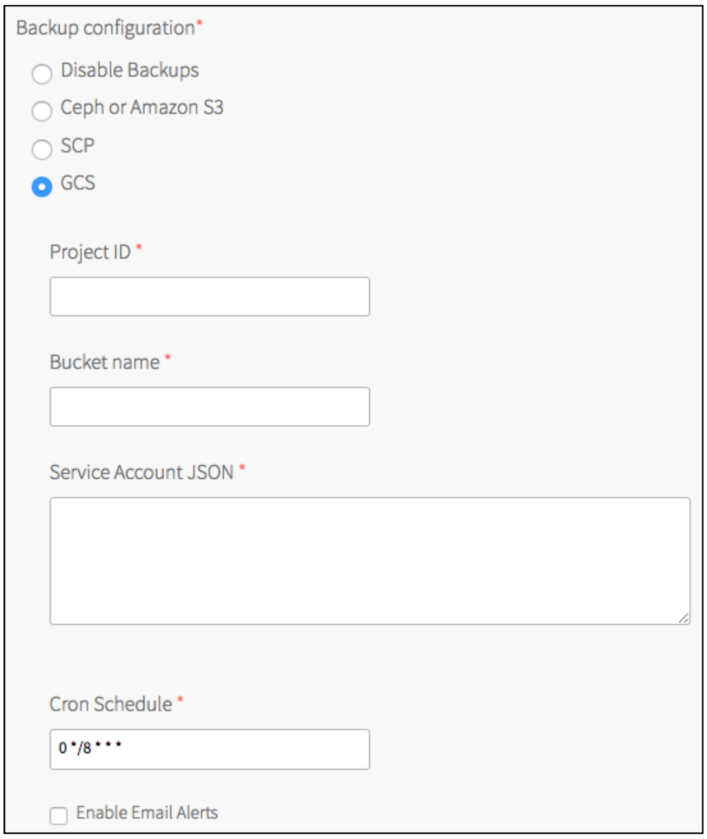
5. Click **Create** and take note of the name and location of the service account JSON file that is downloaded.

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**.



Backup configuration*

Disable Backups
Ceph or Amazon S3
SCP
[✓ GCS]

Project ID *

Bucket name *

Service Account JSON *

Cron Schedule *

0 */8 * * *

Enable Email Alerts

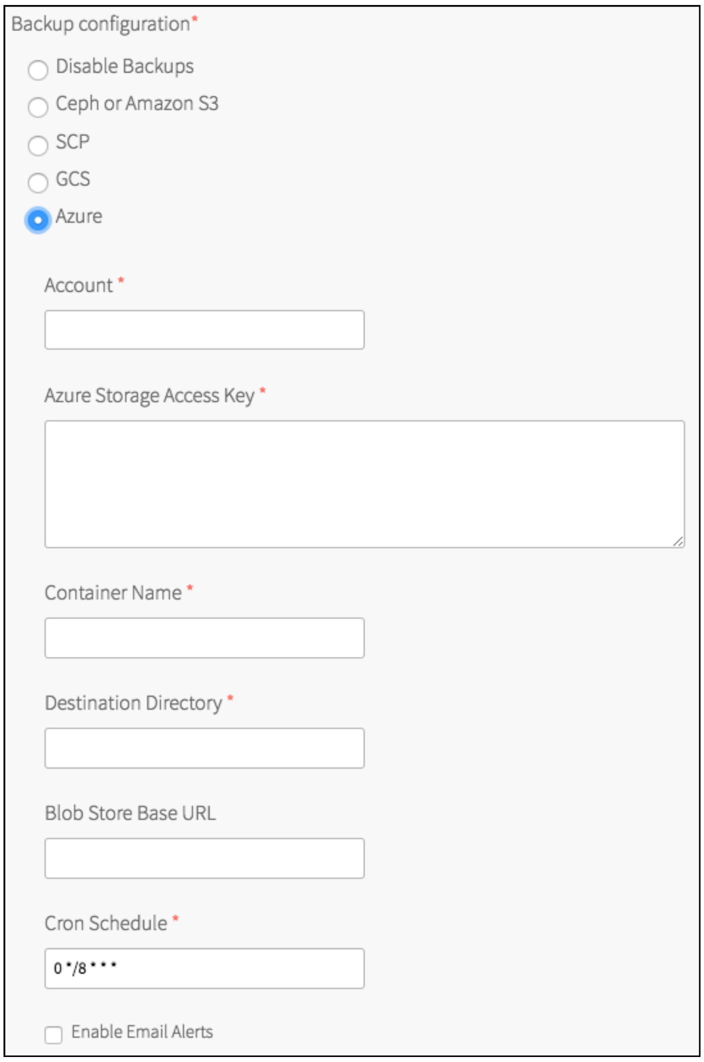
3. Fill in the fields as follows:

   - **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 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*

- Disable Backups
- 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:

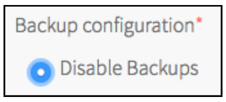
   - **Account**—Enter the Account name for the Microsoft Azure account that you are using.
   - **Azure Storage Access Key**—Enter one of the storage access keys that can be used to access the Azure Storage account.
   - **Container Name**—Enter the container name that backups should be uploaded to.
   - **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 Elastic Runtime, see *Configure Email Notifications* for your IaaS: AWS, Azure, GCP, OpenStack, or vSphere.

Disable Automated Backups

To disable MySQL for PCF automated backups:

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

2. Select **Disable Backups**.



3. Click **Save**.

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

Manual Backup

MySQL for PCF v2.1 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`

   Where:

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

   For example:

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

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

   Where:

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

   For example:

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


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

5. Examine the output and record the following values:

- `hostname` : The MySQL for PCF proxy IP address
- `password` : The password for the user that can be used to perform backups of the service instance database
- `username` : The username for the user that can be used to perform backups of the service instance database

6. Use mysqldump to back up the data for your service instance. Run the following command:

```
mysqldump --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=abcdefghijklm --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:

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-pat
```

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
```

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
```

11. Create the directory structure the tooling expects by creating a `restore-artifact` directory:

```
$ mkdir /var/vcap/store/restore-artifact/
```

12. Move the backup artifact to the script's expected location:

```
$ mv /tmp/mysql-backup-1498863859-0.tar.gpg /var/vcap/store/restore-artifact/
```

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

   ```
   $ /var/vcap/packages/mysql-restore/bin/restore --encryption-key ENCRYPTION-KEY --mysql-username admin --mysql-password ADMIN-PASSWORD
   ```

   where:

   - `ENCRYPTION-KEY` is the `symmetric-key` value from above
   - `ADMIN-PASSWORD` is the `admin_password` value from above

2. Exit the MySQL VM.

3. Determine if the application 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.

Controlling Resource Use

This topic tells operators how to control resource use, that is number and size of service instances used by MySQL for Pivotal Cloud Foundry (PCF) v2, by setting a maximum number of service instances and by only allowing certain orgs to use some service plans.

Example 1: Limit the Number of Service Instances

Resource costs have been increasing. Employees don't delete their databases after tests. To stop employees wasting so many resources, you set quotas for the number of services instances that can exist at a time:

- Set a *global* quota for the total number of service instances used by MySQL for PCF.
- Set a *per-plan* quota for the service-plan with the most expensive service instances, that is, VM types.

Example 2: Limit Plan Access

You have a service plan called db-large which uses large service instances, and it's in an expensive availability zone, too. You want this MySQL for PCF plan used only for your flagship production app. To prevent your development team from creating service instances in this plan, specify that only members of the production org can access the db-large plan.

Quotas

MySQL for PCF v2 offers two types of quotas:

- Global quota
- Service-plan quotas

Global Quota

The global quota limits the *total* number of service instances that can be deployed by MySQL for PCF.

Configuring quotas allows you to limit the number of service instances that can be deployed, thereby ensuring that resource consumption stays within limits. By default, MySQL for PCF ships with a maximum limit of 50 service instances: this means that the MySQL for PCF service broker never deploys more than 50 service instances. You can set this quota to less than 50, but never to more than 50.

To configure a global quota, see Configure Global Quotas.

Service-Plan Quotas

A service-plan quota limits the number of service instances for *one plan*.

In the case where a service plan is configured to use a lot of resources, RAM or disk space, you might want to limit the number of service instances of this plan that can be configured. Setting a service plan quota on the more expensive plan ensures that resource consumption does not exceed an acceptable amount.

To configure a service-plan quota, see Configure Active Service Plans.

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-aee6-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	MySQL Server is currently responding to requests, which indicates if the component is available. **Use**: If the server does not emit heartbeats, it is offline. **Origin**: Doppler/Firehose **Type**: boolean **Frequency**: 30 s
Recommended measurement	Average over last 5 minutes
Recommended alert thresholds	**Yellow warning**: N/A **Red critical**: < 1
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	The percentage of disk used on the persistent and ephemeral file systems. **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. **Origin**: Doppler/Firehose **Type**: Percent **Frequency**: 30 s (default)
Recommended measurement	Max of persistent disk used of all of nodes and Maximum ephemeral disk used of all nodes
Recommended alert thresholds	**Yellow warning**: > 40% **Red critical**: > 45%
Recommended response	Upgrade the service instance to a plan with larger disk capacity.

Connections

/p.mysql/net/connections	
Description	The rate of connections to the server, shown as connections per second. **Use:** If the number of connections drastically changes or if apps are unable to connect, there might be a network or app issue. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30 s
Recommended measurement	(Number of connections / Max connections) over last 1 minute
Recommended alert thresholds	**Yellow warning**: > 80 **Red critical**: > 90
Recommended response	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 application instances and app utilization. If this metric is met or exceeded for an extended period of time, monitor app usage to ensure everything is behaving as expected.

Questions

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

- To send BOSH HM metrics through the Firehose, install the open-source HM Forwarder.
- To retrieve BOSH health metrics outside of the Firehose, install the JMX Bridge for PCF tile.

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	Persistent disk being consumed by the MySQL service instance. **Use:** If the persistent disk fills up, MySQL will be unable to process queries and recovery is difficult. **Origin**: JMX Bridge or BOSH HM **Type**: percent **Frequency**: 60 s (default)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: > 75 **Red critical**: > 90
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	RAM being consumed by the MySQL service instance. **Use:** MySQL increases its memory usage as the data set increases. This is normal, as much of that RAM is used to buffer IO. As long as there is enough remaining RAM for other processes on the instance, the MySQL server should be OK. **Origin**: JMX Bridge or BOSH HM **Type**: percentage **Frequency**: 60 s (default)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: > 95 **Red critical**: > 99
Recommended response	Update the service instance to a plan with more RAM.

CPU

	system.cpu.percent
Description	CPU time being consumed by the MySQL service. **Use:** A node that experiences context switching or high CPU usage will become unresponsive. This also affects the ability of the node to report metrics. **Origin**: JMX Bridge or BOSH HM **Type**: percent **Frequency**: 60 s (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
`/p.mysql/performance/created_tmp_tables`	The number of internal temporary tables created each second by the server while executing statements.	tables/second

Data Source	Description	Metric Unit
`/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 `COM_PING` and `COM_STATISTICS` .	count
`/p.mysql/performance/queries_delta`	The change in the `/performance/queries` 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 `.frm` 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

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 IaaSes such as vSphere have similar resurrection or HA features.
  - Apps cannot connect to MySQL until the VM is recreated and the My SQL server process is resumed.
  - Based on prior experience with BOSH Resurrector, typical downtime is 8-10 minutes.
- **Required Actions:** When the VM comes back, no further action should be 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

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 can run the upgrade-all service broker errand to upgrade existing service instances to be in sync with the new version of the service.

Both operators and developers can upgrade existing service instances individually by running the `cf update-service` command. See Managing Service Instances with the cf CLI.

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

In this topic:

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

Start here if you’re responding to a 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.

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 the following:

```
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
```

1. If the BOSH error shows a problem with the deployment manifest:

    a. Download the manifest for the on-demand service instance by running: `bosh download manifest service-instance_SERVICE-INSTANCE-GUID MY-SERVICE.yml.`

    b. Check the manifest for configuration errors.

    Note: This error does not apply if you are using BOSH CLI v2. In that case, to troubleshoot possible problems with the manifest, open it in a text editor and inspect the manifest there.

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 one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh task TASK-ID`
1.11	`bosh2 task TASK-ID`
1.12 and later	`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
```

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 one of these commands depending on your Ops Manager version:

        - 1.10 and earlier: `bosh tasks`

- 1.11: `bosh2 tasks`
- 1.12 and later: `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. Type `cf service MY-INSTANCE --guid`. This confirms that the the MySQL for PCF service instance exists in BOSH and CF, and returns a GUID.

2. Using the GUID obtained above, run one of the following BOSH CLI commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh vms service-instance_GUID`
1.11	`bosh2 -d service-instance_GUID vms`
1.12 and later	`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 infor
```

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 PCF Elastic Runtime tile is deployed to has network access to the service network. You can find the network definition for this service network in the Ops Manager 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 ERT 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
    b. Run `cf nozzle -f ValueMetric | grep --line-buffered "on-demand-broker/MY-SERVICE"` to find logs from your service in the `cf nozzle` output.

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.

Troubleshooting Components

Guidance on checking for and fixing issues in on-demand service components.

BOSH Problems

Missing BOSH Director UUID

Note: This error does not occur if you are using BOSH CLI v2

If using the BOSH CLI v1, re-add the `director_uuid` to the manifest:

1. Run `bosh status --uuid` and record the `director_uuid` value from the output.

2. Edit the manifest and add the `director_uuid: DIRECTOR-UUID` from the last step at the top of the manifest.

For more, see Deployment Identification in the BOSH docs.

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 Elastic Runtime tile.

Networking

Common issues include:

1. Network latency when connecting to the MySQL for PCF service instance to create or delete a binding.
    - Solution: Try again or improve network performance
2. Network firewall rules are blocking connections from the MySQL for PCF service broker to the service instance.
    - Solution: 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.
3. Network firewall rules are blocking connections from the service network to the BOSH director network.
    - Solution: Ensure that service instances can access the Director so that the BOSH agents can report in.
4. Apps cannot access the service network.

    - Solution: Configure Cloud Foundry application security groups to allow runtime access to the service network.

5. Problems accessing BOSH's UAA or the BOSH director.

    - Solution: Follow network troubleshooting and check that the BOSH director is online.

Validate Service Broker Connectivity to Service Instances

1. To validate you can `bosh2 ssh` onto the MySQL for PCF service broker:

    - **With BOSH CLI v2:** Target the deployment, and reach the service instance.
    - **With BOSH CLI v1:** Download the broker manifest and target the deployment, then try to reach the service instance.

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.
6. 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 one of the following commands:

Ops Manager Version	BOSH Command
1.10 or earlier	`bosh vms --vitals service-instance_GUID`
1.11	`bosh2 -d service-instance_GUID vms --vitals`
1.12 and later	`bosh -d service-instance_GUID vms --vitals`

For additional information, run one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh instances --ps --vitals`
1.11	`bosh2 instances --ps --vitals`
1.12 and later	`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 one of the following commands, depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh deployments`
1.11	`bosh2 deployments`
1.12 and later	`bosh deployments`

2. **For BOSH CLI v1 only:**

   a. Run `bosh download manifest ODB-DEPLOYMENT-NAME odb.yml` to download the ODB manifest.
   b. Select the ODB deployment using `bosh deployment odb.yml`.

3. View VMs in the deployment using one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh instances`
1.11	`bosh2 -d DEPLOYMENT-NAME instances`
1.12 and later	`bosh -d DEPLOYMENT-NAME instances`

4. SSH onto the VM by running one of the following commands:

Ops Manager Version	BOSH Command

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh ssh service-instance_GUID`
1.11	`bosh2 -d service-instance_GUID ssh`
1.12 and later	`bosh -d service-instance_GUID ssh`

5. Download the broker logs by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh logs service-instance_GUID`
1.11	`bosh2 -d service-instance_GUID logs`
1.12 and later	`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 cf CLI command `cf service MY-SERVICE --guid`.

2. **For BOSH CLI v1 only:**

    a. Run `bosh status --uuid` to retrieve the BOSH Director GUID.

    Note: "GUID" and "UUID" mean the same thing.

    b. To download your BOSH manifest for the service, run `bosh download manifest service-instance_BOSH-DIRECTOR-GUID MANIFEST.yml` using the GUID you just obtained and a filename you want to save the manifest as.

    c. Edit the following line in the service instance manifest that you just saved, to include the current BOSH Director GUID:

    ```
    director_uuid: BOSH-DIRECTOR-GUID
    ```

    d. Run `bosh deployment MANIFEST.yml` to select the deployment using the Director UUID.

3. View VMs in the deployment using one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh instances`
1.11	`bosh2 -d DEPLOYMENT-NAME instances`
1.12 and later	`bosh -d DEPLOYMENT-NAME instances`

4. SSH onto a VM by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh ssh service-instance_GUID`
1.11	`bosh2 -d service-instance_GUID ssh`
1.12 and later	`bosh -d service-instance_GUID ssh`

5. Download the instance logs by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh logs service-instance_GUID`
1.11	`bosh2 -d service-instance_GUID logs`
1.12 and later	`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 errands:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command:
   `bosh deployment BOSH_MANIFEST.yml`

2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh run errand ERRAND_NAME`
1.11	`bosh2 -d DEPLOYMENT_NAME run-errand ERRAND_NAME`
1.12 and later	`bosh -d DEPLOYMENT_NAME run-errand ERRAND_NAME`

   Examples:
   `bosh run errand deregister-broker`
   `bosh2 -d DEPLOYMENT-NAME 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, do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command:
   `bosh deployment BOSH_MANIFEST.yml`

2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh run errand register-broker`
1.11	`bosh2 -d DEPLOYMENT-NAME run-errand register-broker`
1.12 and later	`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, do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running the command: `bosh deployment BROKER_MANIFEST.yml` .
2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh run errand deregister-broker`
1.11	`bosh2 -d DEPLOYMENT-NAME run-errand deregister-broker`
1.12 and later	`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 may want to upgrade all the MySQL for PCF service instances to the latest software/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 serially:
  - Issues an upgrade command to the on-demand broker.
  - Re-generates 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. Run the errand by following either of the procedures below.

To run the errand, you can either select the errand through the Ops Manager UI and have it run when you click `Apply Changes` , or do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command: `bosh deployment BOSH_MANIFEST.yml`
2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh run errand upgrade-all-service-instances`
1.11	`bosh2 -d DEPLOYMENT-NAME run-errand upgrade-all-service-instances`

Ops Manager Version	BOSH Command
1.12 and later	`bosh -d DEPLOYMENT-NAME run-errand upgrade-all-service-instances`

Delete All Service Instances

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

Orphan BOSH deployments don't correspond to a known service instance. While rare, orphan deployments can occur. Use the `orphan-deployments` errand to identify them.

The errand does the following:

- Unbinds all applications from the service instances.
- Deletes all service instances sequentially.
- Checks if any instances have been created while the errand was running.
- If newly-created instances are detected, the errand fails.

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, do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running the command: `bosh deployment BROKER_MANIFEST.yml` .

2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh run errand delete-all-service-instances`
1.11	`bosh2 -d service-instance_GUID delete-deployment`
1.12 and later	`bosh -d service-instance_GUID delete-deployment`

Detect Orphaned Instances 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, do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running the command: `bosh deployment BROKER_MANIFEST.yml`

2. Run the errand using one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh run errand orphan-deployments`
1.11	`bosh2 -d DEPLOYMENT-NAME run-errand orphan-deployments`
1.12 and later	`bosh -d DEPLOYMENT-NAME run-errand orphan-deployments`

If orphan deployments exist, the errand script will:

- 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 i



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 lo
  "logs": {
    "blobstore_id": "d830c4bf-8086-4bc2-8c1d-54d3a3c6d88d"
  }
}
```

If no orphan deployments exist, the errand script will:

- 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 it will:

- 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.

Ops Manager Version	BOSH Command
1.10 and earlier	`bosh delete deployment service-instance_SERVICE-INSTANCE-GUID`
1.11	`bosh2 delete-deployment service-instance_SERVICE-INSTANCE-GUID`
1.12 and later	`bosh delete-deployment service-instance_SERVICE-INSTANCE-GUID`

Select the BOSH Deployment for a Service Instance

This is an additional troubleshooting option for **BOSH CLI v1 only**. It does not apply to the BOSH CLI v2.

1. Retrieve the GUID of your service instance with the command `cf service YOUR-SERVICE-INSTANCE --guid`.

2. To download your BOSH manifest for the service, run `bosh download manifest service-instance_SERVICE-INSTANCE-GUID myservice.yml` using the GUID you just obtained and a file name you want to use when saving the manifest.

3. Run `bosh deployment MY-SERVICE.yml` to select the deployment.

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

To retrieve the admin and read-only admin credentials for a service instance, perform the following steps:

1. Identify the service deployment by GUID.

2. Log into BOSH.

3. Download the manifest for the service instance and add the GUID if using the BOSH CLI v1.

   Skip this step if you are using the BOSH CLI v2. You cannot download the manifest with the BOSH CLI v2. Open it in a text editor instead.

4. Look in the manifest for the `admin` and `roadmin` 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

BOSH CLI v2: Viewing statistics

To view usage statistics for any service do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command: `bosh deployment BOSH_MANIFEST.yml`

2. Run the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Commands
v1.10 and earlier	Run the BOSH CLI v1 command `bosh vms --vitals`. To view process-level information, run `bosh instances --ps`.
v1.11	Run the BOSH CLI v2 command `bosh2 -d DEPLOYMENT-NAME vms --vitals`. To view process-level information, run `bosh2 -d DEPLOYMENT-NAME instances --ps`
v1.12 and later	Run the BOSH CLI v2 command `bosh -d DEPLOYMENT-NAME vms --vitals`. To view process-level information, run `bosh2 -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
`on-demand-broker/SERVICE-NAME-MARKETPLACE/quota_remaining`	global quota remaining for all instances across all plans
`on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/quota_remaining`	quota remaining for a particular plan
`on-demand-broker/SERVICE-NAME-MARKETPLACE/total_instances`	total instances created across all plans
`on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/total_instances`	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 support ticket here. Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

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

Pivotal

Using MySQL for PCF

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

These procedures 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, see Managing Service Instances with the cf CLI.

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 into the org and space containing your app

The Create-Bind Process

Because every app and service in PCF is scoped to a space, an app can only use a service if an instance of the service exists in the same space.

To use MySQL in a PCF app:

1. Use the cf CLI or Apps Manager to log into the org and space that contains the app.
2. Make sure an instance of the MySQL for PCF service exists in the same space as the app.
    - If the space does not already have a MySQL for PCF instance, create one.
    - If the space already has a MySQL for PCF instance, you can bind your app to the existing instance or create a new instance to bind to your app.
3. Bind the app to the MySQL for PCF service instance, to enable the app to use MySQL.

Confirm Service Availability

For an app to use a service, 1) the service must be available in the Marketplace for its space and 2) an instance of the service must exist in its space.

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

1. To find out if a MySQL for PCF v2 service is available in the Marketplace:

    a. Enter `cf marketplace`
    b. If the output lists `p.mysql` in the `service` column, on-demand MySQL for PCF 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
[...]
```

2. To confirm that a MySQL for PCF v2 instance is running in the space

a. Enter `cf services`
b. Any `p.mysql` listings in the `service` column are service instances of on-demand MySQL 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, Use, and Manage Service Instances

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, run `cf create-service`:

1. Enter `cf create-service p.mysql db-small SERVICE-INSTANCE`
   Where `SERVICE-INSTANCE` is a name you choose to identify the service instance. This name will appear under `service` [sic] 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 it to a service instance. Do this after you push or re-push the app using `cf push`.

To bind an app to a MySQL instance run `$ cf bind-service`.

1. Enter `cf bind-service APP SERVICE-INSTANCE`
   Where `APP` is the app you want to use the MySQL service instance and `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.

For how to code your app to use MySQL messaging, see **About Using Pivotal MySQL** > **Client Documentation** in the MySQL documentation.

Update to a Larger Service Instance

As apps and their databases grow, it may be necessary to update the service instance to a larger plan.

1. Enter `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
```

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.

Unbind a Service Instance to Your App

To stop an app from using a service it no longer needs, unbind it from the service instance using `cf unbind-service`.

1. Enter `cf unbind-service APP SERVICE-INSTANCE`
   Where `APP` is the app you want to stop using the MySQL service instance and `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

To delete a service instance, run `cf delete-service`.

1. Enter `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.

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

Example Application

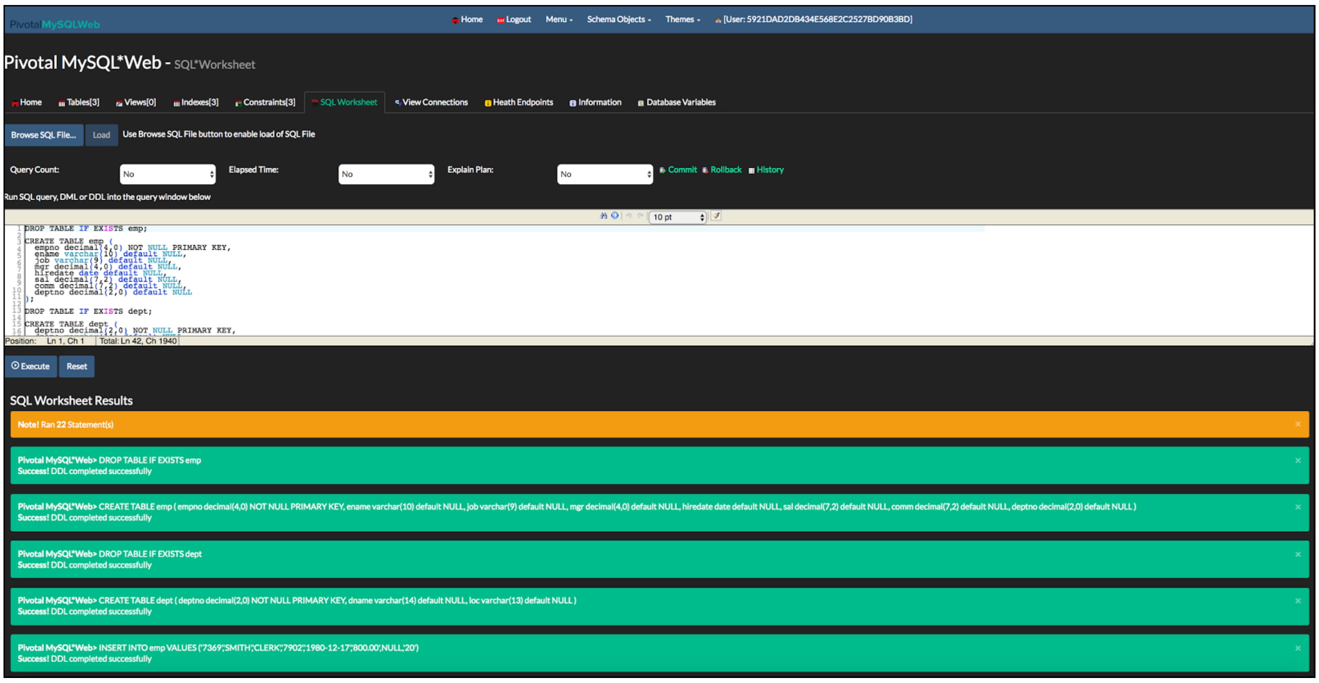
To help application developers get started with MySQL for PCF, we have provided an example application, which can be downloaded here. Instructions can be found in the included README.

MySQL for PCF Tools

The following tools let developers 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.



```
DROP TABLE IF EXISTS emp;

CREATE TABLE emp (
  empno decimal(4,0) NOT NULL PRIMARY KEY,
  ename varchar(10) default NULL,
  job varchar(9) default NULL,
  mgr decimal(4,0) default NULL,
  hiredate date default NULL,
  sal decimal(7,2) default NULL,
  comm decimal(7,2) default NULL,
  deptno decimal(2,0) default NULL
);

DROP TABLE IF EXISTS dept;

CREATE TABLE dept (
  deptno decimal(2,0) NOT NULL PRIMARY KEY,
```

```
Note! Ran 22 Statement(s)

Pivotal MySQL*Web> DROP TABLE IF EXISTS emp
Success! DDL completed successfully

Pivotal MySQL*Web> CREATE TABLE emp ( empno decimal(4,0) NOT NULL PRIMARY KEY, ename varchar(10) default NULL, job varchar(9) default NULL, mgr decimal(4,0) default NULL, hiredate date default NULL, sal decimal(7,2) default NULL, comm decimal(7,2) default NULL, deptno decimal(2,0) default NULL )
Success! DDL completed successfully

Pivotal MySQL*Web> DROP TABLE IF EXISTS dept
Success! DDL completed successfully

Pivotal MySQL*Web> CREATE TABLE dept ( deptno decimal(2,0) NOT NULL PRIMARY KEY, dname varchar(14) default NULL, loc varchar(13) default NULL )
Success! DDL completed successfully

Pivotal MySQL*Web> INSERT INTO emp VALUES ('7369','SMITH','CLERK','7902','1980-12-17','800.00',NULL,'20')
Success! DDL completed successfully
```

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.

Command-Line Interface MySQL Plugin

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

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

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

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

For more information, see the cf-mysql-plugin repository.

Customizing MySQL for PCF

This topic provides instructions for developers customizing their MySQL for PCF service instances.

These procedures 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.

Configuring

MySQL for PCF service instances are configured by default with industry best practices. For more specific use cases, MySQL for PCF service instances can be more customized via the cf CLI.

Provide Optional Parameters

You can create service instance by passing optional parameters to `cf create-service` using the `-c` flag. The `-c` flag accepts a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file.

The MySQL service broker supports the following parameters:

Key	Type	Default	Description
`enable_lower_case_table_names`	Boolean	false	When set to true, sets MySQL's `lower_case_table_names` property to `1`. For more information, consult the MySQL documentation. May only be set when creating a service instance.

If you encounter an error using optional parameters, see Troubleshooting Instances

Migrating a Database to MySQL for PCF v2

This page provides instructions on how to migrate your MySQL for PCF v1.x databases to MySQL for PCF v2. To perform the migration, you use the backup and restore process as follows:

1. Preemptively create a new v2 service instance to minimize downtime.
2. Stop traffic to the old database and take a backup.
3. Import the backup into your new service instance.
4. Connect your app to the new database and restart the app.

Prerequisites

To minimize downtime while migrating, preemptively create a new service instance.

- The application source code can use `p.mysql` as a valid key for accessing the database.
- Create a new service instance of `p.mysql` with a persistent disk large enough for the data. If you do not have an instance, follow the instructions to create a new service instance in the same space as your existing service instance.

> Do not bind the new service instance to your application. We'll do that later.

Back Up Your Current Database

If you are migrating from a MySQL for PCF v1.x database, please refer to the instructions below. If you are migrating from another MySQL provider, please refer to their documentation on taking a MySQL backup.

Back Up Your MySQL for PCF v1.x Instance

1. Stop any traffic to your existing database by stopping the application:

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

2. Unbind your application from the old service instance:

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

3. Use a cf application to create an ssh tunnel to connect to your old service instance. The tunnel may appear as though it is hanging.

> Since your application is now stopped, you'll need another application for the ssh tunnel. If you don't have a running app, you can clone and cf push this sample app for tunneling. You can use any running app in the same Org/Space

   - An ssh tunnel is necessary as the MySQL service is only accessible within the Cloud Foundry network.

4. Take a backup of the existing database in a separate shell:

```
$ mysqldump --single-transaction --add-drop-table --add-locks --create-options --disable-keys --extended-insert --quick --set-charset --routines --flush-privileges -u USERNAME -p -h 0 -
```

   - DB_NAME is the "name" property from your service-key
   - USERNAME is the "username" property from your service-key
   - 63306 is the available port you configured in the ssh tunnel on your local machine
   - You will be prompted for a password after entering the previous command which is the "password" property from your service-key

5. Kill the ssh tunnel

Restore to Your New Database

This section provides steps to restore your backup to a new service instance of MySQL for PCF:

1. Use cf ssh to create a tunnel to remotely access your new MySQL instance.

2. In another shell, restore your backup to the new service instance via the ssh tunnel:

   ```
   $ mysql -u USERNAME -p -h 0 -P 63306 -D service_instance_db < backup.sql
   ```

   - USERNAME is the “username” property from your service-key
   - 63306 is the available port you configured in the ssh tunnel on your local machine
   - backup.sql is the path to the backup on your local machine
   - You will be prompted for a password after entering the previous command which is the “password” property from your service-key

3. Validate that data that you expect to see has been imported into the new service instance by using the following command to enter a mysql shell.

   ```
   $ mysql -u USERNAME -p -h 0 -P 63306 -D service_instance_db
   ```

   - USERNAME is the “username” property from your service-key
   - 63306 is the available port you configured in the ssh tunnel on your local machine
   - You will be prompted for a password after entering the previous command which is the “password” property from your service-key

4. Exit the mysql shell and kill the ssh tunnel

5. Bind the application to the new service instance

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

6. Restage the app to restart the app with the new changes

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

7. Your application is now using your new MySQL for Pivotal Cloud Foundry v2 database 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=z9z6eskdbs1rhtxt",
      "name": "service_instance_db",
      "password": "z9z6eskdbs1rhtxt",
      "port": 3306,
      "uri": "mysql://fefcbe8360854a18a7994b870e7b0bf5:z9z6eskdbs1rhtxt@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 application is only modified when the application 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 Instances

This topic provides basic instructions for app developers troubleshooting On-Demand MySQL for PCF.

Temporary Outages

MySQL for PCF service instances can become temporarily inaccessible during upgrades and VM or network failures. See Service Interruptions for more information.

Errors

You may 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`

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.

Retrieve Service Instance Information

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. Run `cf service SERVICE-INSTANCE-NAME` to retrieve more information about a specific instance.

4. Run `cf service SERVICE-INSTANCE-NAME --guid` to retrieve the GUID of the instance, which is useful for debugging.

Select the BOSH Deployment for a Service Instance

This is an additional troubleshooting option for **BOSH CLI v1 only**. It does not apply to the BOSH CLI v2.

1. Retrieve the GUID of your service instance with the command `cf service YOUR-SERVICE-INSTANCE --guid`.

2. To download your BOSH manifest for the service, run `bosh download manifest service-instance_SERVICE-INSTANCE-GUID myservice.yml` using the GUID you just obtained and a file name you want to use when saving the manifest.

3. Run `bosh deployment MY-SERVICE.yml` to select the deployment.

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 help expedite troubleshooting, if possible also 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.